Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku
Odsek za softversko inženjerstvo

Predmet: Praktikum iz programiranja 2 (13S111PP2)
Predmetni nastavnik: Marko Mišić
Predmetni saradnik: Tamara Šekularac
Školska godina: 2018/2019.

Projektni zadaci za samostalni rad

Verzija dokumenta: 2.6

Uvod

Ispit PP2 se polaže kroz domaće zadatke (5) i završni (ispitni) zadatak. Na domaćim zadacima, student može da osvoji oko 70% od ukupnog broja poena, a ostatak na završnom zadatku. Studenti poseduju različit nivo znanja i razumevanja materije koja se obrađuje u predmetu PP2. Samim tim postoji problem određivanja primerene težine zadataka, tako da slabijim studentima ne budu preteški, a boljim dosadni.

U cilju prevazilaženja pomenutog problema, uvodi se mogućnost da zainteresovani studenti mogu da (neke ili sve) poene koje bi dobili na domaćim zadacima osvoje putem projekta. Projekti se rade samostalno ili u grupama. U nastavku ovog dokumenta se predlaže način sprovođenja ovakvog načina ocenjivanja.

Ciljevi

Uvođenjem projekata kao načina osvajanja poena za ispit iz PP2, očekuje se sledeće:

- motivisanje studenata da kroz izradu projekta nauče više i steknu veća praktična znanja nego što bi to ostvarili kroz izradu domaćih zadataka
- rad i saradnja studenata u manjim grupama (2-5 studenata), uz redovno praćenje, nadgledanje i upravljanje od strane demonstratora i predmetnih nastavnika (neka vrsta mentorstva)
- priprema studenata za razna takmičenja iz informatike

Projekat

Projekat je zamišljen u vidu složenog zadatka ili problema koji studenti treba da realizuju pisanjem programa u programskom jeziku C, a koji od studenata zahteva razmišljanje, čitanje literature i dokumentacije, pretraživanje Interneta, itd. Projekat može da bude pravljenje programa za vođenje evidencije o aktivnostima studenata (poput fakultetskog servisa EVIDES, ali naravno jednostavnije), pravljenje programa za šifrovanje i dešifrovanje teksta, ali može da bude i pravljenje programa za zabavu (na primer jednostavna video-igra). Kompletna realizacija projekta podrazumeva da studenti napišu dokumentaciju o korišćenju programa i dokumentaciju za programere (User's and Programmer's Manual), kao i da dizajniraju i implementiraju intuitivan korisnički interfejs.

Ocenjivanje

Po završetku projekta, studenti brane svoj rad demonstrirajući kako funkcioniše, uz odgovaranje na pitanja predmetnih nastavnika koji utvrđuje da li progam funkcioniše prema zadatoj specifikaciji. Studenti zajedno demonstriraju rad izrađene aplikacije, a zatim pojedinačno odgovaraju na pitanja asistenta o delovima aplikacije koji su oni realizovali. Svaki student koji radi na projektu mora da poznaje sve opšte crte programa koji je realizovan. Broj poena koji može da se osvoji izradom projekta se kreće između 0 i 21.

Komunikacija

Studenti program rade u timu, ali prema zadatoj specifikaciji i podeli posla za svaki projektni zadatak. Kako su delovi programa koje izrađuju različiti studenti najčešće međuzavisni, članovi tima moraju da odgovorno i na vreme realizuju svoje delove rešenja i stavljaju ih na uvid drugima. U tom smislu, studenti treba da koriste odgovarajuće alate za praćenje različitih verzija programa poput SVN (Subversion) ili Git koje studenti mogu pronaći na internetu. Pored samih članova tima, studenti treba da dodaju i predmetne nastavnike i rukovodioca projekta kao članove repozitorijuma. Takođe, strateške odluke koje se tiču opšte strukture programa, relevantnih struktura podataka i slično, studenti u timu treba da donesu zajedno, poželjno u saradnji sa zaduženim rukovodiocem projekta.

Svaki tim redovno prati i nadgleda zaduženi predmetni demonstrator i predmetni nastavnik. Predmetni demonstrator je odgovoran za konsultacije i savetovanje studenata u vezi rešavanja postavljenog problema i u saradnji sa predmetnim asistentom razrešava nedoumice u postavci zadatka. Predmetni demonstrator ne rešava direktno postavljene probleme niti realizuje delove koda. Za komunikaciju sa predmetnim demonstratorom i

nastavnikom se koristi elektronska pošta, a po potrebi se zakazuju i sastanci uživo. Ukoliko postoji problem, najpre se kontaktira predmetni demonstrator, a tek ukoliko problem ne može da se reši, nastavnik.

Studenti su dužni da pišu kratke nedeljne izveštaje o trenutnom napretku i problemima sa kojima su se susreli prilikom izrade projekta. Ukoliko su studenti napisali i neki deo koda, izveštaj treba da sadrži i arhivu sa odgovarajućim .h i .c datotekama. Izveštaj treba da bude kratak i jasan i da pruža trenutni uvid u napredak rada na projektu. Izveštaj šalje jedan od studenata u ime cele grupe, na elektronsku poštu zaduženog demonstratora i predmetnog nastavnika najkasnije do ponedeljka uveče u 23 časova, svake radne nedelje od početka izrade projekta. Očekuje se da tokom izrade projekta studenti napišu bar tri ovakva kratka izveštaja. Izostanak svakog nedeljnog izveštaja povlači oduzimanje po 1 poena od ukupnog skora svakog studenta prilikom konačnog ocenjivanja projekta.

Korisnički interfejs

U zavisnosti od specifikacije zadatka i postavljenog problema, rešenje treba da ima intuitivan i jednostavan korisnički interfejs. Interfejs može biti realizovan putem menija ili konzole. Ukoliko problem zahteva iscrtavanje, sva iscrtavanja se moraju izvršavati na istom (statičkom) ekranu, a ne putem skrolujućeg ekrana.

Dokumentacija

U sklopu projektnog zadatka, studenti su dužni da napišu prateću dokumentaciju. Potrebno je odvojeno napistati uputstvo za upotrebu programa, koje ilustruje sve tipične slučajeve korišćenja, mogućnosti programa, preduslove za korišćenje i slično, i uputstvo za programere koje kratko opisuje način rešavanja programa, korišćene alate i dokumentuje korišćene strukture i funkcije. Dokumentacija se piše prema odgovarajućem šablonu koji se može naći na sajtu predmeta.

Testiranje

Za svaki projektni zadatak, studenti su dužni da pripreme odgovarajući skup test primera kojima će biti testirane funkcionalnosti programa.

Stil pisanja programa

Studenti bi tokom izrade projektnog zadatka trebali da obrate pažnju na stil i način pisanja programa. Očekuje se da napisani kod bude pregledan, uredan i komentarisan na mestima koja su teža za razumevanje. Takođe, očekuje se da programski sistem bude na pogodan način dekomponovan u odgovarajuće .h i .c datoteke. Gde god je pogodno, očekuje se grupisanje promenljivih i korišćenje odgovarajućih struktura podataka.

Opšta napomena

Ukoliko u projektnom zadatku nešto nije dovoljno precizno definisano ili su postavljeni kontradiktorni zahtevi, studenti treba da uvedu razumne pretpostavke, da ih temeljno obrazlože i da nastave da izgrađuju preostali deo svog rešenja na temeljima uvedenih pretpostavki. Zahtevi su ponekad namerno nedovoljno detaljni, jer se od studenata očekuje kreativnost i profesionalni pristup u rešavanju praktičnih problema!

Projektni zadatak 1: Igra „Sudoku“

Autor: Branko Kokanović
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Sudoku sastavljač i rešavač. Sudoku je popularan zadatak (mozgalica) koji se sastoji od inicijalno delimično popunjene tabele (A) od 9x9 polja, a tabela se može posmatrati kao kolekcija manjih tabela (B), dimenzija 3x3 polja, smeštenih u 3 vrste, u svakoj vrsti po 3 tabele. U svakom polju tabele (A) se nalazi najviše jedan jednocifren pozitivan broj, uz ograničenje da u jednoj vrsti i jednoj koloni tabele ne sme da se pojavi isti broj više puta. Brojevi ne smeju da se ponavljaju u okviu tabela (B). Cilj je popuniti sva polja tabele, poštujući ova ograničenja. Program treba da pomogne korisniku da reši zadati zadatak ili da mu pomogne u sastavljanju novog zadatka.
- Raspodela aktivnosti
    - **1. student:** Glavni program, meni za interaktivan rad i sastavljanje novog zadatka**.**
      Program prikazuje meni za interaktivan rad sa korisnikom koji nudi sledeće opcije:
        - pomoć: objašnjenje svih mogućnosti programa
        - učitavanje Sudoku zadatka iz fajla
        - snimanje Sudoku zadatka u fajl
        - brisanje tekućeg zadatka
        - sastavljanje novog zadatka
        - prikazivanje tekućeg izgleda zadatka
        - rešavanje tekućeg zadatka
        - snimanje tekućeg zadatka u SVG fajl
        - kraj rada

      Student treba da osmisli format fajla u kojem će se čuvati i iz kojeg će se čitati podaci o jednom Sodoku zadatku. U saradnji sa ostalim članovima grupe, student treba da osmisli i realizuje odgovarajuću strukturu podataka (i prateće funkcije za njeno manipulisanje) za efikasno pronalaženje rešenja zadatka. U svakom trenutku je potrebno da se zna koja polja su popunjena inicijalnim (zadatim) vrednostima, a koja su popunjena rešavanjem. Ove podatke treba čuvati u fajlu, a prilikom prikazivanja na ekranu treba drugacije oznaciti ta polja. Prilikom snimanja tekućeg zadatka u SVG fajl, korisniku treba omogućiti da bira željeni broj poznatih polja (81 = potpuno popunjena tabela, tj. rešen zadatak) i na osnovu toga reguliše nivo težine zadatka. Student treba da potraži na Internetu informacije os SVG formatu.
      Prilikom zadavanja novog zadatka, korisnik polazi od prazne tabele u koju dodaje vrednosti. Nakon dodavanja svake vrednosti, treba proveriti da li postoji narušavanje zadatih pravila. Takođe, nakon dodavanja vrednosti, korisniku treba omogućiti da pokrene rešavanje, da bi proverio da li je zadatak rešiv.

    - **2. student:** Implementira osnovne algoritme rešavanja Sudoku zadataka
      Student treba da implementira sledeće metode: metoda eliminacije (singles), singlton metoda (hidden singles) i metoda golih parova (naked pairs). O ovim algoritmima student treba da se informiše na Internetu, a o njihovoj implementaciji će biti reči u interakciji studenta sa rukovodiocem projekta. Student je takođe zadužen za realizaciju dela programa koji poziva gorenavedene algoritme. Ukoliko se rešenje ne nađe ni posle primene svih algoritama, korisniku

se saopštava poruka o neuspehu i preporučuje mu se da pokrene algoritam rešavanja tzv. "grubom silom". Treba meriti vreme trajanja ovih algoritama rešavanja Sodoku zadataka.

3. student: Implementira algoritam rešavanja "grubom silom".

Potrebno je realizovati funkciju koja će rešiti zadatak, počevši od zadatog stanja, na dva načina:

- tako što će isprobavati sve moguće kombinacije koje nisu u koliziji sa pravilima igre
- ubacivati jedan po jedan broj-kandidat u tabelu i pozivati algoritme koje je realizovao student 2, dok se ne dođe do rešenja ili iscrpe sve mogućnosti.

O detaljima implementacije ovog algoritma će biti reči u interakciji studenta sa postavljenim rukovodiocem projekta.

Projektni zadatak 2: Simulacija Lengtonovog mrava

Autor: Konstantin Đorđević, Strahinja Milovanović, Natalija Radić
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Simulacija dvodimenzionalne Tjuringove mašine koja radi po pravilima *Lengtonovog mrava*. Potrebno je realizovati grafički prikaz mreže po kojoj se kreće mrav (karakterna grafika), korisnički interfejs koji omogućava upravljanje simulacijom, kao i samu logiku iste, odnosno ponašanje mrava. Univerzum Lengtonovog mrava sastoji se od dvodimenzionalne ravni u obliku kvadratne mreže, u kojoj se kreće mrav u jednom od četiri pravca, kvadrat po kvadrat. Pomeranje se vrši po zadatim pravilima. Na početku, cela mreža je bezbojna, te mrav svojim kretanjem boji polja. Korisnik određuje pravila kretanja mrava tako što, za svaku boju na koju mrav može da naiđe, određuje na koje od susednih polja će mrav skrenuti, kao i novu boju za trenutno polje. Smenjivanje boja u jednom polju se dešava ciklično.
- Raspodela aktivnosti
  - **1. student**: Grafički korisnički interfejs
    Student dizajnira i implementira grafički korisnički interfejs koji omogućava kontrolu nad simulacijom. Upravljanje simulacijom se odvija kroz interaktivni meni. Krajnji korisnik će imati mogućnost upravljanja veličinom prikaza mreže, kao i brzinom kretanja mrava po poljima (*timescale* simulacije). Na intuitivan način će mu biti omogućen i odabir pravila i boja simulacije. Student je zadužen i da realizuje učitavanje i čuvanje simulacije u eksternom binarnom fajlu. Student treba da osmisli format fajla i podatke koji se čuvaju. Podrazumevana podešavanja simulacije treba čuvati u tekstualnoj konfiguracionom fajlu čiji format takođe treba osmisliti. Promenu konfiguracionog fajla omogućiti unutar programa. Ukoliko konfiguracioni fajl ne postoji, koristiti podrazumevane vrednosti koje su fiksirane u programu.
  - **2. student**: Grafički prikaz simulacije i strukture podataka
    Student realizuje iscrtavanje obojene mreže na ekranu, koja u realnom vremenu prikazuje stanje simulacije, koristeći karakternu grafiku. Student je dužan da obezbedi automatsko skaliranje i pomeranje prikaza u zavisnosti od veličine iskorišćenog dela mreže i pozicije mrava. Mreža se iscrtava na osnovu interne memorijske reprezentacije, koju zajedno sa studentom 3 definiše ovaj student. Student takođe realizuje osnovne operacije za dohvatanje i modifikovanje podataka u strukturi i sl.
  - **3. student:** Logika simulacije
    Student implementira celokupnu logiku simulacije, što podrazumeva održavanje i popunjeavanje potrebnih struktura za mrežu i pravila kretanja, te da omogući da se simulacija izvršava beskonačno, tj. dok je korisnik ne prekine. U saradnji sa studentom 2 definiše internu memorijsku reprezentaciju problema. Ovaj student obezbeđuje funkcije kojima povezuje korisničke kontrole (na kojima radi student 1) sa grafičkim izlazom programa (koji realizuje student 2).

Projektni zadatak 3: Igra „Pacman“

Autor: Mina Šekularac, Bogdan Bebić, Uroš Krstić
Rukovodilac projekta: Bogdan Bebić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Potrebno je realizovati popularnu arkadnu igru Pacman. Igra će biti namenjena samo za jednog igrača (*singleplayer*). Potrebno je implementirati grafički prikaz, logiku (*engine*) same igre i veštačku inteligenciju duhova. U okviru igre, igrač se kreće kroz lavirint koji sadrži Pac-tačke (*Pac-Dots*) i četiri duha koji mogu da ubiju igrača. Cilj igre je da igrač prikupi sve Pac-tačke i tako završi odgovarajući nivo. Igrač može tokom igre da sakupi određene bonus dodatke koji utiču na tok igre i broj sakupljenih poena.
- Raspodela aktivnosti
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika

    Zadatak ovog studenta je realizacija svih grafičkih elemenata igre. Student treba da napravi interaktivni meni u kome se može pokrenuti nova igra, nastaviti igra, mogu podesiti parametri same igre (težina duhova, odabir duhova, broj života i sl.) i pogledati lista najboljih skorova. Potrebno je obezbediti sve funkcije koje će se koristiti pri iscrtavanju samog nivoa, a koje će pozivati 2. student. Listu najboljih skorova bi trebalo zaštiti od neovlašćene promene van igre (zaštita od „starijeg brata/sestre“).
  - **2. student:** Implementira logiku igre

    Ovaj student implementira odgovarajuću strukturu podataka za generisanje i smeštanje mape i elemenata na mapi. Zadatak ovog studenta je da dohvata komande igrača, komande duhova (koje obezbeđuje 3. student) i da ažurira stanje i pozove funkcije 1. studenta radi iscrtavanja novog stanja na konzoli. Logika igre obuhvata i analizu trenutnog stanja mape (efekti pojačanja, vođenje računa o trenutnom skoru i broju života i sl.). Takođe, ovaj student je zadužen za implementaciju nastavljanja igre na osnovu funkcija koje implementiraju druga dva studenta.s
  - **3. student:** Implementira računarskog protivnika

    Zadatak studenta je da implemetira algoritme koji upravljaju računarskim protivnicima, kao i samim igračem za demonstracioni mod igre. Potrebno je napisati funkcije koje odlučuju o akcijama duhova u narednom koraku (kretanje u jednom od četiri smera ili mirovanje) na osnovu trenutnog stanja na mapi i njihovog odabira (različiti tipovi duhova se biraju u meniju). Potrebno je napraviti bar četiri duha sa različitim algoritmima za kretanje.

Projektni zadatak 4: Igra „Snake battle“
Autor: Nikola Bebić
Rukovodilac projekta: Nikola Bebić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Igra „Snake battle“ je bazirana klasičnoj Snake game igri, sa nekoliko izmena. Mapa je pravougaona i nema zidove. Na mapi se na početku nalazi najviše 4 zmije. Cilj igre je biti poslednja preživela zmija, a zmija umire tako što udari u ivicu mape, u drugu zmiju, ili u sebe. Zmiju može da kontroliše ili čovek, ili računar. Najviše dva čoveka mogu kontrolišu zmiju u jednoj partiji, i to jedan strelicama a drugi tasterima „WASD“. Moguće je i pokrenuti partiju u demonstracionom režimu, kada samo računari igraju jedan protiv drugog. Svaka zmija neprekidno raste, odnosno rep joj je fiksiran, a glava joj se pomera na jednu od 4 strane. Jedan potez se odigrava tako što, ciklično, unapred određenim redosledom, svaka zmija pomera glavu za jedno polje u neku stranu. Ukoliko zmija umre, igrač koji je kontroliše gubi partiju, a ta zmija se uklanja sa mape, tako da ostale mogu da popune mesta koja je do sad zauzimala. Treba meriti vreme svake partije, i ukoliko je neka partija u najkraćih 10 vremena, traži se unos imena igrača i igrač se upisuje u listu koja se čuva u datoteci koja treba da bude zaštićena od izmena van programa.
- Raspodela aktivnosti
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika
    Student dizajnira i implementira grafički korisnički interfejs koji omogućava kontrolu nad igrom. Glavni program sadrži interaktivni meni sa opcijama nova igra, opcije, o igri, najbolji skorovi, autori i izlazak. U opcijama se podešava broj igrača, broj računarskih zmija, brzina zmija, veličina mape, težina svakog od računarskih protivnika, brisanje tabele najboljih skorova, izbor boja i slično. Kada se odabere nova igra, zmijama se dodele redni brojevi, i početne pozicije na mapi.
  - **2. student:** Implementira logiku igre
    Ovaj student implementira odgovarajuću strukturu podataka za smeštanje mape i elemenata na mapi. Student realizuje algoritam za precizno merenje vremena. Takođe bi trebalo da osmisli način za pomeranje zmije uz pomoć tastera tastature (kada igrač unosi izbor, karakter izbora se ne sme videti, i ne mora se pritisnuti enter za prihvatanje znaka), i pomoću funkcija koje će realizovati treći student. Potrebno je i implementirati algoritam koji proverava da li je neka zmija završila partiju, da li se sama igra završila, i ažurira stanje mape.
  - **3. student:** Implementira računarskog protivnika
    Zadatak studenta je da implemetira algoritam koji će igrati protiv ostalih igrača. Treba implementirati bar dva nivoa težine (lak i težak). Po završetku igre, ukoliko je ostvaren rezultat koji je u 10 najboljih, igraču se traži da unese ime i rezultat se čuva u datoteku što implementira treći student. Datoteka bi trebalo da bude binarnog tipa, uz zaštitu od neovlašćene promene van igre.

Projektni zadatak 5: Igra “Ultimate Tic Tac Toe”

Autori: Kristijan Žiža, Jovan Đukić, Tamara Šekularac
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Implementira modifikaciju igre X-OX sa karakternom grafikom. Prvi igrač stavlja X gde god želi. Svaki naredni potez mora biti odigran u velikom polju koje odgovara malom polju u kome je odigran poslednji potez. Ako takvog polja nema, igra se u centralnom simetričnom polju, pa ako je i to polje zauzeto, onda bilo gde. Igrač koji spoji tri najmanja polja u malom polju pobeđuje na tom malom polju. Igrač koji pobedi u tri mala polja (prema pravilima X-OX) u velikom polju (celoj matrici) dobija partiju. U posebnoj datoteci *high_score.txt* se vodi računa o deset najboljih igrača protiv računarskog protivnika. Pamtiti ime igrača, datum, kao i broj poteza koji je doveo do pobede. Igrače rangirati prema broju poteza koji su načinili do pobede.
-
- najboljih deset vremena rešavanja igre, imenima igrača i datumu rešavanja. Studenti zajedno osmišljavaju najbolju strukturu za predstavljanje nivoa u memoriji i njegovu obradu.
- Raspodela aktivnosti
    - **1. student**: Grafički interfejs i PvP.
      Zadatak ovog studenta je realizacija svih grafičkih elemenata igre. Student treba da napravi interaktivni meni u kome se može pokrenuti nova igra, nastaviti sačuvana stara igra, podesiti parametri same igre (težina i sl.) i pogledati lista najboljih skorova. Potrebno je obezbediti sve funkcije koje će se koristiti pri iscrtavanju samog nivoa a koje će pozivati 2. student. Takođe, potrebno je obezbediti komunikaciju izmedju igrača i računara, kao i pregled trenutnog stanja igre. Takođe, zadatak ovog studenta je i da obezbedi mogućnost da igranja igre od strane dva igrača. Datoteku u kojoj se nalazi lista najboljih skorova potrebno je zaštiti od neautorizovane promene van igre (tzv. *zaštita od mlađeg brata*).
    - **2. student:** Logika igre i demonstracioni režim.
      Student treba da implementira pravila igre, da prihvata komande igrača, proverava ispravnost poteza, proverava da li je igra završena, ažurira trenutno stanje. Takodje, zadatak ovog studenta je i implementacija demonstracionog režima igre u kome dva računarska protivnika igraju jedan protiv drugog.
    - **3. student**: Veštačka inteligencija.
      Potrebno je napisati funkcije koje na osnovu trenutnog stanja igre, definišu poteze koji će računar odigrati kada je on na potezu. Zadatak studenta je da „nauči“ računar da igra što efikasnije. Potrebno je obezbediti tri nivoa težine igranja: lako, srednje i teško. Takođe, potrebno je obezbediti igraču opciju za pomoć (*hint*) koja predlaže narednih nekoliko poteza koje igrač treba da povuče da bi došao do pobede.

Projektni zadatak 6: Muzička biblioteka

Autor: Miloš Tijanić, Dušan Jovanović
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Treba realizovati aplikaciju koja omogućava laku pretragu i održavanje MP3 datoteka na disku. Aplikacija treba da poseduje svoje datoteke u kojima će da čuva informacije o MP3 datotekama iz direktorijuma koje je zadao korisnik. Korisniku treba da bude omogućena i izmena ID3 tag-ova datoteka iz svoje biblioteke. Takođe, treba omogućiti snimanje pretrage ili izabranih datoteka u plejlistu odgovarajućeg tipa.
- Raspodela aktivnosti
  - **1. student**: Grafički interfejs i rad sa greškama
    Grafički interfejs ove aplikacije treba da sadrži okvir u kome će korisnik moći da pregleda svoju biblioteku sortiranu po raznim kriterijumima. Takođe, treba da postoji način da se korisniku prikazuju samo deo biblioteke koji odgovara kriterijumu pretrage koji korisnik zada. Ostale funkcije koje interfejs treba da pruži korisniku su i dodavanje novog direktorijuma u biblioteku, dodavanje pojedinačnih fajlova, izbacivanje fajlova iz biblioteke, osvežavanje biblioteke, ručna izmena ID3 tag-a određenog fajla. Ovaj student takođe treba da osmisli, realizuje i stavi na raspolaganje za korišćenje ostalima sistem za obradu grešaka do kojih može doći prilikom rada aplikacije. Takođe, treba realizovati mogućnost snimanja rezultata pretrage u odgovarajući format plejliste (PLS, M3U ili ASX).
  - **2. student**: Rad sa fajl sistemom, čitanje MP3 formata i rad sa ID3
    Zadatak ovog studenta je da ostalima pruži funkcije koje se tiču rada sa fajl sistemom kao što su dohvatanje svih datoteka jednog direktorijuma, otvaranje i zatvaranje fajlova, brisanje fajlova, dodavanje fajlova, rad sa internim fajlovima i njihovo održavanje. Drugi skup funkcija se odnosi na rad sa samim MP3 formatom i čitanjem ID3 tag-a i njegovom izmenom. Takođe, ovaj student je zadužen za implementiranje bilo koji internih struktura koje će se koristiti za komunikaciju između interfejsa i biblioteke (zadaci studenta 1. i 3. respektivno).
  - **3. student:** Formiranje biblioteke i obrada pretrage
    Biblioteka treba da se sastoji od jednog ili više fajlova koji će se interno nalaziti u aplikaciji i treba da sadrži informacije koje će omogućiti brzu pretragu i jednostavnu izmenu MP3 fajlova i njihovih tag-ova. Na ovom studentu je da nađe algoritme i strukture podataka koji najviše odgovaraju ovom problemu i da ih implementira radi njegovog rešavanja. Potrebno je na određeni način optimizovati algoritme pretrage i strukture podataka tako da vreme odziva prilikom pretrage bude kratko. Cela biblioteka se ne sme čuvati sve vreme u radnoj memoriji računara i na ovom studentu je da taj mehanizam rada i obezbedi.

PRILOG: Formati plejlista

Sa povećanjem procesorske snage sa jedne, i multimedijalnih mogućnosti računara sa druge strane, pojava zapisa sa visokim kvalitetom zvuka bila je više nego očekivana. Tržište medijuma je ispratilo nove potrebe tržišta računara novim formatima koji pružaju veliki kapacitet za smeštanje podataka (razne varijante CD, DVD i slično). Razvoj raznih formata kompresije sa minimalnim gubicima je još više olakšao reprodukciju zvučnih zapisa na kućnim računarima. Zbog velike količine zapisa, potrebno je organizovati ih radi reprodukcije po željenom redosledu. Datoteka koja sadrži osnovne podatke o numerama koje treba reprodukovati i redosledu u žargonu se naziva **plej-lista** (engl. *playlist*).

Prikazani formati mogu imati i složeniju strukturu od one koja je predstavljena ovde, ali sve formate treba koristiti onako kako su ovde opisani. Više informacija o prikazanim formatima možete naći na:

http://en.wikipedia.org/wiki/M3U
http://en.wikipedia.org/wiki/PLS_(file_format)
http://en.wikipedia.org/wiki/Advanced_Stream_Redirector

Sva tri tipa formata mogu biti napravljena uz pomoć programa KMPlayer, koji može biti preuzet sa adrese http://kmplayer.en.softonic.com/, dok M3U i PLS liste koristi i program WinAmp, koji može biti preuzet sa adrese http://www.winamp.com/. Liste koje kreiraju KMPlayer ili WinAmp se po formatu mogu minimalno razlikovati od formata koji je prikazan u ovom fajlu. Zato će možda biti potrebno, pre testiranja programa sa rešenjem DZ5, nekim editorom teksta prilagoditi plej-listu formatu opisanom u ovom dokumentu.

M3U

```
#EXTM3U
#EXTINF:302,Chris Rea - Looking For The Summer
C:\muzika\Chris Rea - Looking For The Summer.mp3
#EXTINF:265,Simply Red - Holding Back The Years
C:\muzika\Simply Red - Holding Back The Years.mp3
#EXTINF:207,Mambo Kings - Luz de luna
C:\muzika\Mambo Kings - Luz de luna.mp3
```

Prva linija označava da se radi o proširenom formatu M3U liste i ona je ista u svakoj plej-listi. U nastavku je opis numera koje se nalaze u plejlisti. Svaka numera je prestavljena sa dva reda. Prvi red počinje sa **`#EXTINF:`**, nakon čega slede celi broj koji predstavlja trajanje numere u sekundama i naziv numere koji će biti prikazan u programu za reprodukciju. Druga linija predstavlja putanju do datoteke sa zvučnim zapisom.

PLS

```
[playlist]
NumberOfEntries=3
File1=C:\muzika\Chris Rea - Looking For The Summer.mp3
Title1=Chris Rea - Looking For The Summer
Length1=302
File2=C:\muzika\Simply Red - Holding Back The Years.mp3
Title2=Simply Red - Holding Back The Years
Length2=265
File3=C:\muzika\Mambo Kings - Luz de luna.mp3
Title3=Mambo Kings - Luz de luna
Length3=207
Version=2
```

Prva linija uvek ima sadržaj **`[playlist]`**. Druga linija govori koliko numera ima u plej-listi i ima format **`NumberOfEntries=broj numera`**. U nastavku sledi opis numera koje su u plej-listi. Svaka numera je opisana sa tri reda. Prvi ima format **`File#=putanja do fajla`**, drugi ima format **`Title#=naziv numere`**, a treći ima format **`Length#=trajanje numere u sekundama`**. Znak **`#`** označava redni broj numere koja se opisuje. Fajl se završava jednom linijom u formatu **`Version=broj verzije`**, koja govori o verziji PLS fajla (samo verzija 2 je značajna za projekat).

ASX

```
<Asx Version = "3.0" >

<Entry>
<Title>"Chris Rea - Looking For The Summer.mp3"</Title>
<Ref href = "C:\muzika\Chris Rea - Looking For The Summer.mp3"/>
</Entry>

<Entry>
<Title>"Simply Red - Holding Back The Years.mp3"</Title>
<Ref href = "C:\muzika\Simply Red - Holding Back The Years.mp3"/>
</Entry>

<Entry>
<Title>"Mambo Kings - Luz de luna.mp3"</Title>
<Ref href = "C:\muzika\Mambo Kings - Luz de luna.mp3"/>
</Entry>

</Asx>
```

ASX format je napisan na jeziku XML. Sve informacije o numerama su smeštene u tag **Asx** koji ima atribut **Version** sa vrednošću **3.0**. Informacije o numeri su smeštene u tag <**Entry**>. Naziv numere, koji će biti prikazan u plejeru, čuva se u tagu **Title** a putanja do fajla se čuva u tagu **Ref**, tačnije njegovom atributu **href**.

Projektni zadatak 7: Igra “Super Mario”

Autor: Bogdan Bebić
Rukovodilac projekta: Bogdan Bebić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Potrebno je realizovati uprošćenu verziju popularne igre Super Mario. Igra će biti namenjena samo za jednog igrača (*singleplayer*). Potrebno je implementirati grafički prikaz, logiku (*engine*) same igre, generisanje nivoa i veštačku inteligenciju protivnika. U okviru igre, igrač preuzima ulogu italijanskog vodoinstalatera koji se kreće kroz nivo koji sadrži protivnike, prepreke i pojačanja. Cilj je da prođe kroz kraljevstvo gljiva, pobedi kralja kornjača Bauzera (*Bowser*) i oslobodi princezu Breskvicu (*Peach*).
- Raspodela aktivnosti
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika

    Zadatak ovog studenta je realizacija svih grafičkih elemenata igre. Student treba da napravi interaktivni meni u kome se može pokrenuti nova igra, nastaviti igra, mogu podesiti parametri same igre (teveličina mape, broj života i sl) i pogledati lista najboljih skorova. Potrebno je obezbediti sve funkcije koje će se koristiti pri iscrtavanju samog nivoa, a koje će pozivati 2. student.

  - **2. student:** Implementira logiku igre

    Ovaj student implementira odgovarajuću strukturu podataka za smeštanje mape i elemenata na mapi. Zadatak ovog studenta je da dohvata komande igrača, komande protivnika (koje obezbeđuje 3. student) i da ažurira stanje i pozove funkcije 1. studenta radi iscrtavanja novog stanja na konzoli. Logika igre obuhvata i analizu trenutnog stanja mape (efekti pojačanja, vođenje računa o trenutnom skoru, broju života i sl). Takođe, ovaj student je zadužen za implementaciju nastavljanja igre na osnovu funkcija koje implementiraju druga dva studenta.

  - **3. student:** Implementira generisanje mape i računarske protivnike

    Zadatak studenta je da implemetira generisanje mape i algoritme koji upravljaju računarskim protivnicima. Potrebno je napisati funkcije koje odlučuju o akcijama protivnika u narednom koraku (kretanje u jednom od dva smera ili mirovanje) na osnovu trenutnog stanja na mapi. Listu najboljih skorova bi trebalo zaštiti od neovlašćene promene van igre.

Projektni zadatak 8: *Game of Life* simulacij

Autor: Filip Živković
Rukovodilac projekta: ?
Broj studenata: 2

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: jednostavna simulacija celularnog automata koji se ponaša po pravilima igre *Game of Life*. Potrebno je realizovati kreiranje novog praznog sistema zadatih dimenzija, snimanje i učitavanje sa diska, iscrtavanje u različitim rezolucijama i navigaciju po ekranu. Sve poznate konfiguracije pamtiti u posebnom logu: pamti se iteracija i koja konfiguracija je u pitanju. Predvideti postojanje više od jedne vrste nepraznih ćelija.
- Raspodela aktivnosti
    - **1. student: Interfejs**
      Potrebno je sistem prikazati korisniku grafički što reprezentativnije. Korisnik može da napravi novi sistem ili pusti simulaciju trenutnog sistema proizvoljnom brzinom (iscrtavanje na svaku, na svaku drugu, na svaku desetu iteraciju itd). Predvideti da korisnik može da snimi deo ili celu konfiguraciju u fajl na disku, da može da učita celu konfiguraciju ili deo konfiguracije na određeno mesto u sistemu. Korisnik može da izmeni stanje proizvoljne ćelije. Omogućiti kretanje prozora za prikaz po sistemu (ukoliko je sistem veći od polja za prikaz). Omogućiti više različitih rezolucija za prikaz. Pretpostaviti da može da postoji više od jedne vrste ne praznih ćelija (detalji u prilogu). Obezbediti merenje i ispis vremena simulacije.

    - **2. student: Simulacija**
      Ovaj student pravi internu simulaciju sistema. Predvideti proizvoljne dimenzije sistema, (uključujući beskonačno). Potrebno je detektovati i pribeležiti postojanje svake od karakterističnih konfiguracija (detaljnije u prilogu).

PRILOG: malo više o *Game of Life*

Osnovna pravila *Game of Life*:
- Automat se sastoji od ćelija raspoređenih u kvadratnu mrežu
- Svaka ćelija je u jednoj itraciji ili mrtva ili živa
- Svaka živa ćelija sa manje od dva živa suseda (susedi su ćelije koje dele ivicu ili ćošak ¨C svaka ćelija ima 8 suseda) postaje mrtva u sledećoj iteraciji (smrt od usamljenosti)
- Svaka živa ćelija sa preko tri suseda postaje mrtva u sledećoj iteraciji (smrt od prenaseljenosti)
- Svaka mrtva ćelija sa tačno tri suseda postaje živa u sledećoj iteraciji (reprodukcija)
- Ostale ćelije ne menjaju stanje u sledećoj iteraciji

Ukoliko postoji više od jedne vrste živih ćelija, ćelija koja postaje živa (reprodukcijom) je one vrste koje je većina njenih suseda. Potrebno je predvideti sledeće odnose među vrstama ćelija (neka su vrste A i B)
- Koegzistencija: svaka vrsta postoji za sebe i funkcioniše po gornjim pravilima;
- Predator i plen: svaka ćelija prve vrste (plen) čiji sused je ćelija druge vrste (predator) postaje mrtva u sledećoj iteraciji
- Virus koji se širi: svaka ćelija prve iteracije (nezaraženi) čiji sused je ćelija druge vrste (zaražena) menja vrstu u sledećoj iteraciji (postaje zaražena) ukoliko ostane živa u toj iteraciji
- Nepoznatost: kada se računaju susedi prve vrste, ćelije druge vrste se računaju kao mrtvi; kada se računaju susedi druge vrste, ćelije prve vrste se računaju kao mrtve;

Projektni zadatak 9: Alat za kompresiju i dekompresiju podataka

Autor: Grupa studenata / Đurđević Đorđe / Marko Mišić
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Kompresija podataka korišćenjem DEFLATE algoritma, po ugledu na ZIP format. Treba omogućiti kompresiju i dekompresiju podataka pomoću alata sa podrškom za rad iz komandne linije i pomoću interaktivnog menija.
- Raspodela aktivnosti
  - **1. student**: Glavni program koji omogućava interakciju sa korisnikom
    Treba predvideti dva načina rada: pomoću komandne linije i putem interaktivnog menija. U interaktivnom načinu rada se prikazuju meniji sa mogućim operacijama za manipulaciju datotekama koje se kompresuju i podešavanje opcija alata za kompresiju. Treba omogućiti kompresiju više datoteka istovremeno u istu arhivu. Student samostalno definiše odgovarajući format za parametre prilikom rada iz komandne linije u zavisnosti od podržanih opcija programa. Ukoliko se neki parametar ne navede uzima se default vrednost koja je zapisana u konfiguracionom fajlu programa ako taj fajl postoji, odnosno default vrednosti koje su fiksirane u programu u suprotnom. Takođe, treba voditi računa o međusobnoj isključivosti određenih parametara i upozoriti korisnika ako do toga dođe. Treba omogućiti posebnu opciju u meniju za kreiranje i izmenu parametara konfiguracionog fajla. Program mora podržati opciju –h ili –? (*help*), kao i –a (*about*). Prilikom rada iz komandne linije, potrebno je podržati selekciju datoteka za kompresiju pomoću osnovnih džoker znaka (* i ?). Poseban parametar –l treba da omogući pravljenje dnevnika (*log*) celog procesa kompresije i dekompresije. U dnevniku se beleže svi karakteristični koraci prilikom operacija kompresije i dekompresije i trajanje pojedinih delova i celokupne operacije.

  - **2. student**: Dinamički i statički Huffman-ov alagoritam i DEFLATE algoritam
    Zadatak ovog studenta je da realizuje dinamički i statički Huffman-ov alagoritam (koder i dekoder), kao i realizacija dinamičkog Huffman-ovog stabla i metoda za umetanje i čitanje iz njega. Student je dužan da osmisli sve neophodne strukture podataka i stavi ih na raspolaganje ostalim studentima. Ovaj student realizuje DEFLATE algoritam uz pomoć gotovih struktura koje realizuju studenti 2 i 3.

  - **3. student:** LZW algoritam, rad sa datotekama i upravljanje greškama
    Student realizuje LZW algoritam (koder i dekoder), kao i proširenu tabelu simbola i metoda za umetanje i čitanje iz nje. Takođe, student realizuje funkciju za odabir metode pakovanja bloka podataka. Ovaj student je dužan da realizuje funkcije za rad sa datotekama. Student osmišljava i realizuje mehanizam kojim će se vršiti izveštavanje o greškama koje su nastale prilikom izvršenja programa (u vidu celobrojne vrednosti), a koji će svi učesnici u projektu koristiti.

Projektni zadatak 10: DivX titl konvertor

Autor: Marko Mišić
Rukovodilac projekta: ?
Broj studenata: 4

Student	Broj indeksa	e-mail
1.		
2.		
3.		
4.		

- Kratak opis projekta: konvertor DivX prevoda sa podrškom za pomeranje prevoda u vremenu. Podržani formati: SubRip, MicroDVD, Mplayer MPSub (više detalja o formatima u prilogu). Treba omogućiti konverziju iz bilo kog u bilo koji format podržanih prevoda. Treba voditi računa o ispravnim vrednostima pomeraja za prevode i oporavku od grešaka u slučaju neispravnog formata datoteke.
- Raspodela aktivnosti
  - **1. student**: Glavni program koji omogućava interakciju sa korisnikom
    Treba predvideti dva načina rada: pomoću komandne linije i putem interaktivnog menija. U interaktivnom načinu rada se prikazuju meniji sa mogućim operacijama za manipulaciju prevodima (trenutno učitani fajl sa prevodima , ime pod kojim se snima, vrsta konverzije...) i informacije o samom učitanom fajlu (dužina trajanja, format, broj redova, veličinu u bajtovima...). Prilikom rada iz komandne linije, zadaje se ime ulazne i izlazne datoteke, vrsta konverzije, pomeraj i eventualno još neki podaci (npr. broj *fps* ako se radi sa MicroDVD prevodima). Student samostalno definiše odgovarajući format za parametre prilikom rada iz komandne linije. Ukoliko se neki parametar ne navede uzima se default vrednost koja je zapisana u konfiguracionom fajlu programa ako taj fajl postoji, odnosno default vrednosti koje su fiksirane u programu u suprotnom. Takođe, treba voditi računa o međusobnoj isključivosti određenih parametara i upozoriti korisnika ako do toga dođe. Treba omogućiti posebnu opciju u meniju za kreiranje i izmenu parametara konfiguracionog fajla. Program mora podržati opciju –h ili –? (*help*), kao i –a (*about*). Takođe, potrebno je podržati i paketnu (*batch*) obradu, koja se zadaje posebnim parametrom –b:ime_batch_fajla. Prilikom takve obrade, u svakom redu batch tekst fajla će se nalaziti podaci za jednu obradu po istom formatu kao za rad sa komandnom linijom (osim opcije –b koja nije raspoloživa). Potrebno je obezbediti oporavak od grešaka, tako da ako neki fajl nije ispravan treba nastaviti dalje sa obradom sledećeg. U režimu paketne obrade, poseban parametar –l treba da omogući pravljenje dnevnika (*log*) celog procesa obrade. U dnevniku se beleže nazivi neuspešno obrađenih fajlova i razlozi, a za uspešno obrađene fajlove broj obrađenih titlova i ukupnu dužinu svih titlova. Student osmišljava i realizuje mehanizam kojim će se vršiti izveštavanje o greškama koje su nastale prilikom izvršenja programa (u vidu celobrojne vrednosti), a koji će svi učesnici u projektu koristiti.

  - **2. student**: Interna reprezentacija i manipulacija prevodima, i izveštavanje o greškama
    U saradnji sa studentom 3, ovaj student osmišljava i realizuje internu reprezentaciju prevoda za efikasno obrađivanje. Zadatak studenta je da prouči od kojih sve parametara se sastoji jedan prevod i da predloži odgovarajuću strukturu podataka. Nakon odobravanja od strane rukovaoca projekta, student treba da realizuje funkcije potrebne za manipulisanje odabranom strukturom podataka tako da omogući efikasno obrađivanje prevoda. To bi bile funkcije za dodavanje i brisanje (jedne rečenice) prevoda, brisanje prevoda u celini, umetanje prevoda, dohvatanje određenog prevoda, dohvatanje ukupne dužine trajanja prevoda. Treba proveravati da li se nakon dodavanja ili izmene nekog prevoda javlja nekonzistentno stanje (na primer rečenice se vremenski preklapaju). Takođe treba proveravati da li je vreme dodatog ili menjanog prevoda korektno

uneseno (nema negativnih vrednosti, početak nije nakon završetka). Student takođe treba da osmisli i realizuje internu strukturu podataka kojom se šifra greške prevodi u odgovarajuću tekstualnu poruku. Ta struktura podataka u osnovi treba da ima dve funkcije: (1) da joj se prijavi šifra i opis greške i (2) da se od nje zatraži opis greške koji odgovara zadatoj šifri.

- **3. student:** Učitavanje, snimanje i jednostavna obrada prevoda
  Student realizuje učitavanje prevoda iz i snimanje prevoda u sledeće formate: SubRip, MicroDVD, MPlayer MPSub. Ovaj student treba da prouči navedene formate i da pomogne studentu 2 da osmisli internu reprezentaciju prevoda. Nakon toga, ovaj student koristi funkcije koje realizuje student 2. Jednostavne obrade koje ovaj student treba da vrši nad prevodima su sledeće: pretvaranje svih slova u prevodu u mala ili velika, postavljanje prvog slova rečenice na veliko, ispravljanje grešaka u prevodu poput slučajnog CAPS LOCK (**aCCIDENTAL CAPS LOCK USAGE**) ili slučajnih velikih slova (**TWo INitial Caps**) prelamanje prevoda u više redova nakon određenog broja znakova, brisanje prevoda , dodavanje prevoda, promena prevoda (vreme ili tekst)

- **4. student**: Složena obrada prevoda
  Ovaj student treba da realizuje složenu obradu nad prevodima:
  - spajanje dva uzastopna prevoda sa malim vremenskim razmakom i malim brojem znakova u jedan
  - razdvajanje jednog dugotrajnog prevoda sa velikim brojem znakova u više uzastopnih prevoda koji se prikazuju u različitim vremenskim intervalima
  - pomeranje prevoda u vremenu, svih ili u zadatom opsegu
    - skraćenje ili produženje trajanja
    - zadavanje preciznog vremena za početni i krajnji prevod
    - povećavanje ili smanjivanje vremenskog rastojanja između uzastopnih prevoda

    Ako se radi u zadatom opsegu, posebno se naznačava da li se izmena (ekstrapolacijom) primenjuje i na ostatak prevoda
  - nadovezivanje fajlova sa prevodima (jedan na kraj drugog) ili podela jednog fajla sa prevodima na više

  Parametri na osnovu kojih se odlučuje šta je "mali vremenski razmak" ili "mali broj znakova", itd..., se unose preko standardnog ulaza ili iz posebnog konfiguracionog fajla.

PRILOG: opis formata fajlova sa titlovima

Postoji više formata titlova, koji imaju ili nemaju određene mogućnosti (višejezičnost, stil slova u tekstu titla, i slično). Sa aspekta ovog dokumenta, od pomenutih mogućnosti su mnogo bitniji načini zapisa vremena pojavljivanja i uklanjanja titlova. Tri prikazana formata koriste kao parametre apsolutno vreme od početka zapisa, ili relativno vreme od prethodnog titla, ili broj slika od početka zapisa. Osim navedenog, bitna razlika između navedenih formata je i način predstavljanja prelaska teksta u narednu liniju – SubRip i MPlayer koriste standardni '\n' (prelazak u naredni red), dok MicroDVD koristi '|' (engl. *pipe*). Prilikom konverzije između dva formata, treba voditi računa o navedenim razlikama koje mogu postojati u načinu predstavljanja vremena i tekst.

SubRip (.SRT)

Ovaj format karakteriše apsolutno vreme pojavljivanja i uklanjanja titlova, sa preciznošću od 1 milisekunde, uz razdvajanje dva titla jednim praznim redom teksta. Primer za ovaj format sledi:

```
1
00:00:15,000 --> 00:00:18,000
A long, long time ago...

2
00:00:18,000 --> 00:00:21,000
in a galaxy far away...

3
00:00:21,000 --> 00:00:24,000
Naboo was under an attack.

4
00:00:25,000 --> 00:00:27,500
And I thought me and
Qui-Gon Jinn could

5
00:00:27,500 --> 00:00:30,000
talk the Federation into

6
00:00:30,000 --> 00:00:34,000
...maybe cutting them a
little slack.
```

MicroDVD (.SUB, .TXT)

Ovaj format karakteriše redni broj slike (engl. *frame*) u kome titl treba da se pojavi, odnosno ukloni. Zavisno od broja slika u sekundi za dati multimedijalni zapis (engl. *frame per second*, odnosno *fps*), ovaj titl se može pojaviti ranije (fps=25), odnosno kasnije (fps=23.976). Svaki titl je u jednom redu teksta. Primer za ovaj format, koji sadrži isti titl, kao i prethodni, je prikazan ovde, uz pretpostavku da je fps=25:

```
{375}{450}A long, long time ago...
{450}{525}in a galaxy far away...
{525}{600}Naboo was under an attack.
{625}{688}And I thought me and|Qui-Gon Jinn could
{688}{750}talk the Federation into
{750}{850}...maybe cutting them a|little slack.
```

MPlayer (.SUB)

Za razliku od prethodna dva formata, koji vreme uglavnom računaju od početka zapisa, ovaj format ima mogućnost i da računa vreme relativno – svaki titl ima informaciju o vremenu proteklom u odnosu na prethodni titl, i o sopstvenom trajanju. Kao i kod prvog opisanog formata, i ovde je razdvajanje dva titla ostvareno jednim praznim redom teksta. Primer sledi, uz komentare tipične za ovaj tip:

```
# first number  : wait this much after previous subtitle disappeared
# second number : display the current subtitle for this many seconds
15 3
A long, long time ago...

0 3
in a galaxy far away...

0 3
Naboo was under an attack.

1 2.5
And I thought me and
Qui-Gon Jinn could

0 2.5
talk the Federation into

0 4
...maybe cutting them a
little slack.
```

Projektni zadatak 11: Generator HTML galerija

Autor: Đurđević Đorđe / Marko Mišić
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: generator HTML albuma slika na osnovu učitanih imena slika. Slike su raspoređene u tabeli dimenzija MxN. Svakoj slici može da se definiše tekst koji će biti prikazan ispod nje. Korisnik može da interaktivno definiše parametre na osnovu kojih se formira album, putem jednostavnog menija. Takođe, treba omogućiti i ubacivanje proizvoljnog teksta, pre ili nakon generisanog albuma. Tekst može da se unosi sa glavnog ulaza ili iz datoteke. Meni treba da ima najamnje sledeće stavke:
    1. definisanje naslova stranice
    2. definisanje izgleda stranice
    3. dodavanje slika
        3.1 sa standardnog ulaza
        3.2 u vidu imena i opsega brojeva
        3.3 u vidu imena koje sadrži džoker znake
        3.4 iz tekstualnog fajla
    4. uklanjanje slike
    5. definisanje teksta za opis slike
    6. definisanje dimenzija tabele u albumu
    7. kreiranje i snimanje HTML stranice
    8. učitavanja i snimanje trenutno aktivnog projekta
    9. brisanje svih unetih parametara
    10. izlaz
- Raspodela aktivnosti
    - **1 student**: realizuje glavni program kojim korisnik može da interaktivno definiše izgled i sadržaj HTML stranica. Student treba da realizuje tačke 1, 2, 3.1, 3.2, 4, 5, 6, 8 i 9. U tački 2, definiše se izgled stranice tj. boje, fontovi. Parametri se čitaju sa standardnog ulaza ili iz tekstualne datoteke.

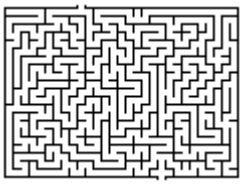
    - **2 student**: realizuje tačku 3.3 i deo tačke 7. U okviru tačke 3.3, student treba da čita sadržaj zadatog direktorijuma, pronađe imena svih fajlova koji odgovaraju zadatom imenu sa džoker znakom i doda ih u spisak slika. U okviru tačke 7, student treba da napravi kostur HTML stranice, na osnovu definisanog izgleda stranice. U slučaju da je uneti broj slika veći od dimenzija tabele, potrebno je kreirati nekoliko HTML stranica među kojima može da se vrši sekvencijalna navigacija (prethodni/sledeći) ili direktna navigacija (1, 2, 3...)

    - **3 student**: realizuje tačku 3.4 i deo tačke 7. i tačku 8. U okviru tačke 3.4, student treba da čita imena fajlova sa slikama iz tekstualnog fajla. U okviru tačke 7, student treba na napravi tabelu sa slikama, za svaku HTML stranicu koju je kreirao student 2. U okviru tačke 8, trebalo bi napraviti odgovarajući tekstualni format za snimanje u učitavanje parametara trenutno aktivnog projekta.

Projektni zadatak 12: Igra „Lavirint“

Autor: Mirko Francuski
Rukovodilac projekta: Nikola Bebić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		



- Kratak opis projekta: Igra „Lavirint“. Ova igra se igra tako što igrač ulazi u lavirint sa jedne strane i pokušava da nađe put kroz njega do izlata na drugoj strani. U slučaju da igrač negde zastane, postoji kompjuterska pomoć (hint) koja pomaže pri odabiru puta. Izgled lavirinta koji se koristi je sledeći: hodnici se seku pod pravim uglom, može sadržati slepe puteve (ćorsokake), postoji samo jedan ulaz i izlaz, ne smeju postojati nedostupna mesta i cirkularni putevi. Takođe, u posebnoj datoteci *high_score.txt* se vodi računa o najboljih deset vremena rešavanja lavirinta, imenima igrača i datumu rešavanja. Studenti zajedno osmišljavaju najbolju strukturu za predstavljanje lavirinta u memoriji i njegovu obradu.
- Raspodela aktivnosti:
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika
    Glavni program sadrži interaktivni meni sa opcijama nova igra, nastavak započete igre i snimanje trenutne, opcije, pomoć, o igri, najbolji skorovi, autori i izlazak. U opcijama se podešava veličina lavirinta, izbor boja za zidove, izbor karaktera za prikazivanje igrača, kao i algoritama za generisanje i rešavanje lavirinta, brisanje tabele najboljih skorova i sl. Kada se odabere nova igra, iscrtava se prazan lavirint definisanih dimenzija (postoje samo spoljašnji zidovi), a zatim se poziva se algoritam za generisanje lavirinta, koji je osmislio drugi student, i lavirint se iscrtava na ekranu na mestu gde se prethodno nalazio prazan lavirint. Omogućiti opciju za iscrtavanje lavirinta po koracima (iteracijama) algoritma za generisanje koje realizuje student 2. Korisnik se postavlja na ulaz lavirinta. Izgled lavirinta podrazumeva korišćenje boja i ASCII tabele kodova. Ulaz u lavirint bi trebalo označiti crvenom, a izlaz zelenom bojom, dok je boja zidova i pozadine korisnički definisana. Pri pozivanju pomoći, iscrtati put od mesta gde se nalazi korisnik do mesta kuda bi trebalo ići ka izlazu lavirinta. Opcija za pomoć treba da prikaže samo nekoliko narednih koraka, ne ceo put ka izlazu. U svakom trenutku treba omogućiti prelazak sa ekrana za igru na ekran sa glavnim menijem i obrnuto. Takođe, student bi trebalo da osmisli način za kretanje korisnika kroz lavirint uz pomoć tastera tastature (kada igrač unosi izbor, karakter izbora se ne sme videti, i ne mora se pritisnuti enter za prihvatanje znaka).

  - **2. student:** Implementira algoritam za pravljenje lavirinta
    Student bi trebalo da napravi generator lavirinta. Prilikom započinjanja nove igre, bira se algoritam za generisanje. Na raspolaganju su sledeći izmenjeni algoritmi:
    – *„Depth-First Search“:*
      1. Izabrati ćeliju i označiti je kao izlaz,
      2. Označiti trenutnu ćeliju kao „posećena“,
      3. Ako ćelija ima neposećenih komšija:
         1) Slučajno odabrati jednog od komšija,
         2) Staviti trenutnu ćeliju na stek,
         3) Skloniti zid između trenutne ćelije i izabrane ćelije,
         4) Izabranu ćeliju napraviti trenutnom ćelijom,
         5) Rekurzivno pozivati ovu funkciju,
      4. Ako nema:

1) Skinuti poslednju ćeliju sa steka,
2) Vratiti se na prethodno pozivanje ove funkcije.

– *Primov algoritam:*

1. Početi sa mrežom zidova,
2. Nasumično izabrati ćeliju, označiti je kao deo lavirnta i dodati sve njene zidove u listu zidova,
3. Dok ima zidova u listi:
    1) Slučajno izabrati zid iz liste. Ako ćelija sa suprotne strane nije u listi, onda:
        1. Napraviti od zida prolaz, i označiti ćeliju sa suprotne strane kao deo lavirinta,
        2. Dodati susedne zidove ćelije u listu zidova.

– *Kruskalov algoritam:*

1. Napraviti listu zidova i skup za svaku ćeliju, tako da svaki skup sadrži samo tu ćeliju,
2. Za svaki zid, na slučajan način:
    1) Ako ćelije odvojene zidom spadaju u odvojene skupove:
        1. Skloniti trenutni zid,
        2. Spojiti dva skupa prethodno razdvojenih ćelija.

Student se ohrabruje da osmisli i implementira neki drugi algoritam za generisanje lavirinta umesto nekog od ponuđenih. Potrebno je meriti vreme pravljenja lavirinta, korišćenjem funkcija koje realizuje student 3.

- **3. student:** <u>Implementira algoritam za pronalaženje puta u lavirintu</u>

  Student bi trebalo da osmisli ili potraži na Internetu algoritam za rešavanje lavirinta. Ulazne vrednosti tog algoritma moraju biti dve ćelije, između kojih se mora pronaći put (ne moraju to biti ulaz i izlaz). Potrebno je realizovati bar tri takva algoritma. Potrebno je postojanje funkcije pomoći, koja za dobijeno mesto u lavirintu vraća put kojim bi trebalo ići, do prvog raskršća, da vodi ka uspešnom rešavanju lavirinta. Potrebno je meriti vreme rešavanja. Ovaj student je zadužen da osmisli i realizuje funkcije za precizno merenje vremena koje će se koristiti u programu. Ovaj student je zadužen i za održavanje liste najboljih skorova u igri, njeno učitavanje i snimanje u datoteku. Listu interno realizovati kao ulančanu listu, a tabelu najboljih skorova je potrebno na neki način zaštiti od neautorizovane promene van same igre.

Projektni zadatak 13: Igra „Potapanje brodova“

Autor: Mirko Francuski
Rukovodilac projekta: ?
Broj studenata: 2

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Igra „Potapanje brodova“. Ova igra se igra izmeću 2 igrača, u ovom slučaju između korisnika i računara ili između dva korisnika.
  Pravila igre:
  - Svakom igraču je dodeljena tabela od 10x10 polja, koja predstavlja oblast mora u kojoj su sakriveni igračevi brodovi. Kolone se obeležavaju slovima abecede od A do J, a vrste celim brojevima od 1 do 10.
  - Svaki igrač na početku bitke raspolaže sledećim brodovima:
    - 1 brod veličine 5 polja,
    - 2 broda veličine 4 polja,
    - 3 broda veličine 3 polja,
    - 4 broda veličine 2 polja.
  - Brodove je moguće orijentisati samo vertikalno i horizontalno (sva polja koja zauzima jedan brod su poravnata na duž jedne linije). Brodovi smeju da se dodiruju. Brodovi su nepokretni u toku igre.
  - Svaki igrač, pre početka igre, raspoređuje svoje brodove, tako da protivnički igrač ne vidi njihov raspored.
  - U toku igre, igrači naizmenično „gađaju“ brodove drugog igrača, svaki u toku svog poteza. Gađanje se obavlja navođenjem koordinate u ranije opisanoj tabeli gde igrač pretpostavlja da se nalaze protivnikovi brodovi. U toku jednog poteza, igrač "gađa" sve dok pogađa protivnikove brodove. Prvi put kada promaši, na red dolazi drugi igrač. Gađanje polja koje je već bilo gađano se tretira kao promašaj. Protivnički igrač je dužan da saopšti da li je brod pogođen, ali ne i da li je potopljen, ukoliko nisu pogođeni svi delovi broda.
  - Igra se završava kada neki od igrača izgubi sve svoje brodove.
- Raspodela aktivnosti:
  - **1. student**: Glavni program, crtanje tabela za igru i ispis informacija

    Glavni program se sastoji iz menija koji se nalazi u kome se nalaze sledeći izbori:
    1. Nova partija,
    2. Hala slavnih,
    3. Opcije,
       1) Promeniti ime igrača,
       2) Snimiti partiju,
       3) Učitati partiju,
       4) Vratiti se nazad,
       5) Igra sa jednim ili dva igrača
       6) Izbor boja za iscrtavanje tabela za igranja i jnihovih elemenata
    4. O igri,
    5. Izlaz.

    Prilikom započinjanja nove partije prelazi se na novi ekran, gde se iscrtavaju dve prazne tabele, svaka za po jednog igrača, i odgovarajuće informacije o tekućoj igri (imena igrača, proteklo vreme u igri, broj potopljenih i preostalih brodova za svakog igrača, broj gađanja i sl.). Zatim svaki korisnik vrši

postavljanje svojih brodova u tabelu. Za brodove različite veličine usvojiti različite znakove za prikaz. Nakon postavljanja brodova, prelazi se gađanje protivničih brodova. Prvi igrač na potezu u igri se bira slučajno. U svakom trenutku treba omogućiti prelazak sa ekrana za igru na ekran sa glavnim menijem i obrnuto. Takođe, student bi trebalo da osmisli način za kretanje korisnika kroz tabelu protivničkog igrača uz pomoć tastera tastature i izbor polja za gađanje. Grafičko okruženje se sastoji u korišćenju boja i ASCII tabele kodova za crtanje tabele, brodova, mesta gađanja (drugačije označavanje za pogodak i promašaj). Ovaj student je zadužen za kompletnu kontrolu toka i završetka igre. U slučaju pobede, korisnik se upisuje u halu slavnih koju realizuje student 2.

- **2. student**: Implementacija računarskog protivnika.
  Student bi trebalo da osmisli takozvanu veštačku inteligenciju, tj. računarskog protivnika. Taj algoritam bi trebalo da radi dve stvari: na početku igre postavi brodove i u toku igre gađa protivničke. Brodovi se postavljaju slučajno, uz izbegavanje ili obavezno postavljanje na neka određena polja, npr. izbegavanje centra ili obavezno postavljanje u neki ugao (na studentu je da odluči i osmisli još neka pravila).
  Algoritam gađanja bi trebalo da radi sledeće:
  - Nasumično gađa (vodeći računa o tome da ne gađa isto mesto dva puta)
  - U slučaju pogotka, na osnovu prethodnih gađanja proceni sledeće mesto na kome će se naći neki deo broda
  - Ponavlja prethodni korak dok se igra ne završi

  Takođe, ovaj student, u saradnji sa drugim članom grupe, bi trebalo da osmisli i realizuje odgovarajuću strukturu podataka (i prateće funkcije za njeno manipulisanje) za efikasno predstavljanje brodova u tabeli. Potrebno je realizovati i opciju za pomoć, koju igrač može da pozove u svakom trenutku i koja treba da daje predlog za sledeći potez. Ovaj student je zadužen i za održavanje liste najboljih skorova u igri (hale slavnih), njeno učitavanje i snimanje u datoteku. Hala slavnih sadrži imena i rezultate (broj gađanja) za 10 najboljih igrača i implementira se kao ulančana lista. Halu slavnih je potrebno na neki način zaštiti od neautorizovane promene van same igre.

Projektni zadatak 14: Igra „Battle Tank“

Autor: Jovan Stupar
Rukovodilac projekta: ?
Broj studenata: 4

Student	Broj indeksa	e-mail
1.		
2.		
3.		
4.		

- Kratak opis projekta: Potrebno je realizovati pojednostavljenu verziju popularne igre *Battle City (Tank 1990)*. Igra će biti namenjena samo za jednog igrača (singleplayer), i koristiće karakternu grafiku. Potrebno je implementirati grafički prikaz, logiku (engine) same igre i veštačku inteligenciju protivnika. Takođe, igra treba da omogući i tzv. demo mod, u kome računarski protivnici igraju jedni protiv drugih.
- Raspodela aktivnosti
  - **1. student:** Grafički interfejs
    Zadatak ovog studenta je realizacija svih grafičkih elemenata igre. Student treba da napravi interaktivni meni u kome se može pokrenuti nova igra, nastaviti sačuvana stara igra, podesiti parametri same igre (težina botova, veličina mape na kojoj se igra i sl.) i pogledati lista najboljih skorova. Potrebno je obezbediti sve funkcije koje će se koristiti pri iscrtavanju samog nivoa a koje će pozivati 2. student.

  - **2. student:** Logika igre
    Zadatak ovog studenta je da dohvata komande igrača, komande botova (koje obezbeđuje 3. student) i da ažurira stanje i pozove funkcije 1. studenta radi iscrtavanja novog stanja na konzoli. Logika igre obuhvata i analizu trenutnog stanja mape (efekti pojačanja, generisanje neprijateljskih tenkova, vođenje računa o trenutnom skoru i sl.). Takođe, ovaj student je zadužen implementacije demo režima igre na osnovu funkcija koje implementiraju druga dva studenta.

  - **3. student:** Veštačka inteligencija:
    Potrebno je napisati funkcije koje odlučuju o akcijama botova u narednom koraku (kretanje u jednom od 4 smera, pucanje na igrača, pucanje na igračevu bazu, ili ne rade ništa) na osnovu trenutnog stanja na mapi i njihove „pameti“ (parametra difficulty koji se podešava u meniju). Potrebno je napraviti barem 3 vrste botova različite veštine (*easy, medium, hard*).

  - **4. student:** Editor i generisanje mapa
    Ovaj student treba da obezbedi generisanje nivoa na osnovu parametara podešenih u meniju i predefinisanih mapa koje se čuvaju u datotekama. Predefinisane mape korisnik pravi pomoću editora nivoa koji ovaj student treba da realizuje. Takođe, dužnost ovog studenta je da osmisli format binarne datoteke za čuvanje generisanih nivoa. Omogućiti automatsko (slučajno generisanje) delova mape (zidovi i fiksne prepreke). Dužnost ovog studenta je i održavanje liste najboljih skorova u igri. Datoteku u kojoj se nalazi lista potrebno je zaštiti od neautorizovane promene van igre (tzv. *zaštita od mlađeg brata*).

Prilog: Pravila igre Battle Tank

Battle City (poznata i pod imenom Tank 1990) je pucačka 2D igrica iz 1985. godine.

Igrač kontroliše jedan tenk

- Tenk se može kretati u četiri pravca i ispaljivati projektil u pravcu u kom se kreće (na stranu na koju je okrenut).

Igrač mora uništiti sve neprijateljske tenkove na nivou

- Neprijateljski tenkovi se pojavljuju na vrhu ekrana u levom i desnom ćošku.

Neprijateljski tenkovi pokušavaju da unište igračevu bazu kao i samog igrača

- Baza predstavlja jedno polje koje se nalazi na dnu ekrana u sredini i okruženo je jednim slojem zida od cigle.

Nivo je pređen kada igrač uništi svih N tenkova

- parametar N se zadaje u opcijama i raste kako nivoi propagiraju.

U datom trenutku na ekranu se može pojaviti najviše N tenkova i igračev tenk.

- parametar N se implicitno određuje u oprijama na osnovu težine same igre

Igra se završava kada je uništena igračeva baza ili je igrač izgubio sve živote. Inače je igra beskonačna.

Postoje dve vrste neprijateljskih tenkova:

- Obični
- Specijalni – nakon što ih igrač uništi za sobom ostavljaju određeno pojačanje

Pojačanja:

- Stvaraju se na mestu gde je uništen specijalni tenk ili nasumično na mapi na određeni interval vremena.
- Postoji više vrsta pojačanja:
    - Dodatni život: uvećava broj života igrača za 1
    - Bomba: uništava sve vidljive neprijateljske tenkove na ekranu
    - Zvezda:
        - 1 zvezda dozvoljava veći fire-rate(brže pucanje)
        - 2 zvezde omogućavaju ispaljivanje 2 metka u isto vreme
        - 3 zvezde omogućavaju igraču da uništava čelične zidove
    - Sat: zamrzava sve neprijateljske tenkove na određeno vreme
    - Lopata: dodaje sloj čeličnog zida oko igračeve baze na određeno vreme
    - Štit: čini igrača neranjivim na određeno vreme.

Polja na mapi:

- Igrač
- Botovi
- Pojačanja
- Projektili
- Zid od cigle (uništiv zid)
- Čelični zid (neuništiv, osim ako igrač nema tri zvezdice)
- Vodeno polje (projektili mogu prolaziti preko njega ali ne i tenkovi)

Projektni zadatak 15: Igra „Bomberman“

Autor: Miloš Tijanić
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Treba realizovati osiromašenu verziju popularne igre *Bomberman*. Igra će biti samo za jednog igrača (singleplayer), i koristiće karakternu grafiku. Potrebno je implementirati logiku (engine) igre, grafički prikaz i veštačku inteligenciju protivnika. Takođe, igra treba da omogući i tzv. demo mod, u kome računarski protivnici igraju jedni protiv drugih.

- Raspodela aktivnosti
  - **1. student**: Grafički interfejs
    Ovaj student realizuje sve grafičke elemente igre. Potrebno je napraviti interaktivni meni u kome se mogu podešavati sve opcije vezane za igru (veličina/oblik nivoa, broj/veština botova, brzina, itd). Pored ovoga, potrebno je obezbediti sve funkcije koje će se koristiti pri iscrtavanju samog nivoa, a koje će student 2. pozivati da prikaže stanje. Ovaj student je zadužen i za održavanje liste najboljih skorova u igri, njeno učitavanje i snimanje u datoteku. Listu interno realizovati kao ulančanu listu, a tabelu najboljih skorova je potrebno na neki način zaštiti od neautorizovane promene van same igre.

  - **2. student**: Logika igre, generator nivoa
    Student implementira celokupnu logiku igre. Potrebno je detektovati komande igrača, dohvatiti komande botova (funkcije koje obezbeđuje student 3), i ažurirati stanje igre. Novo stanje se zatim prikazuje pomoću funkcija koje je obezbedio student 1. Pored ovoga, ovaj student je dužan za realizaciju generatora nivoa, koji će na osnovu nekoliko parametara (dimenzije, stepen popunjenosti) proizvesti slučajan nivo za igranje. Takođe, ovaj student je zadužen implementacije demo režima igre na osnovu funkcija koje implementiraju druga dva studenta.

  - **3. student:** Veštačka inteligencija
    Ovaj student je dužan da napiše funkcije koje odlučuju akcije botova. Ove funkcije kao parametar primaju stanje igre, i vraćaju akciju koju robot bot radi u tom trenutku (kretanje, postavljanje bombe, ništa). Student je dužan da napravi bar tri različita ponašanja botova, koji odgovaraju njihovim stepenima veštine (*easy, medium, hard*).

Prilog: Osnovna pravila Bomberman igre:

Na mapi se mogu naći:

- Agenti – Igrač ili botovi
- Zid – Uništivi ili neuništivi
- Pojačanje – Može se stvoriti nakon uništavanja zida, ima različite efekte
- Bomba – Svaki agent može da postavi bombu, koja otkucava neko vreme pre nego što eksplodira
- Ništa – Ako ništa od navedenog nije na polju, onda se preko njega može slobodno kretati.

Moguće akcije agenta su:

- Kretanje u jednom od četiri smera – Ne može se kretati preko zida, bombe, drugog agenta ili ivice mape. U slučaju da se stane na polje koje sadrži pojačanje, njegov efekat se primenjuje na agenta
- Postavljanje bombe – Na trenutnom polju se postavi bomba kada ga agent napusti.

Moguća pojačanja su:

- Povećanje dometa bombe
- Pojačanje bombe – Mogućnost da uništi neuništive zidove
- Povećanje broja bombi koje agent može da postavi u jednom trenutku – Podrazumevano, agent može imati samo jednu svoju bombu na tabli u svakom trenutku, ovo pojačanje može taj broj da poveća.

Efekat bombe:

- Bomba nakon postavljanja otkucava neko vreme pre detoniranja
- Nakon što detonira, eksplozija putuje u sva četiri pravca dok ne udari u prepreku – Agenta ili zid.
- U slučaju da eksplozija udari agenta, on umire

U slučaju da eksplozija udari zid, taj zid se uništava ukoliko je ili tip zida „uništivi“ ili je jačina bombe povećana da može uništavati i neuništive.

Projektni zadatak 16: Igra „Mummy Maze“

Autor: Nikola Milutinović, Petar Mitrović
Rukovodilac projekta: Nikola Bebić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Igra „Mummy Maze“ je bazirana na lavirintu. Ova igra se igra tako što se igrač nalazi u lavirintu na proizvoljnoj poziciji i pokušava da nađe put kroz njega do izlaza. U lavirintu se nalazi i mumija koja pokušava da uništi igrača i ne dozvoli mu izlazak iz lavirinta. U svakom koraku, igrač može da napravi samo jedan potez, dok mumija pravi dva. U lavirintu može postojati i više mumija. Mumije mogu imati različite nivoe inteligencije. Takođe, u slučaju da igrač negde zastane, postoji kompjuterska pomoć (hint) koja pomaže pri odabiru puta. Izgled lavirinta koji se koristi je sledeći: hodnici se seku pod pravim uglom, može sadržati slepe puteve (ćorsokake), ne postoji ulaz i postoji samo jedan izlaz, ne smeju postojati nedostupna mesta, a smeju cirkularni putevi. U posebnoj datoteci *high_score.txt* se vodi računa o najboljih deset vremena rešavanja igre, imenima igrača i datumu rešavanja. Studenti zajedno osmišljavaju najbolju strukturu za predstavljanje nivoa u memoriji i njegovu obradu.
- Raspodela aktivnosti:
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika
    Glavni program sadrži interaktivni meni sa opcijama nova igra, nastavak započete igre i snimanje trenutne, opcije, pomoć, o igri, najbolji skorovi, autori i izlazak. U opcijama se podešava veličina nivoa, izbor boja za zidove, izbor karaktera za prikazivanje igrača, algoritma za generisanje nivoa, težinu i broj računarskih protivnika, brisanje tabele najboljih skorova i sl. Kada se odabere nova igra, iscrtava se prazan nivo (lavirint) definisanih dimenzija (postoje samo spoljašnji zidovi), a zatim se poziva se algoritam za generisanje nivoa, koji je osmislio drugi student, i nivo se iscrtava na ekranu na mestu gde se prethodno nalazio prazan nivo. Omogućiti opciju za iscrtavanje nivoa po koracima (iteracijama) algoritma za generisanje koje realizuje student 2. Korisnik se postavlja na proizvoljno mesto u lavirintu. Izgled nivoa podrazumeva korišćenje boja i ASCII tabele kodova. Pri pozivanju pomoći, iscrtati put od mesta gde se nalazi korisnik do mesta kuda bi trebalo ići ka izlazu lavirinta. Opcija za pomoć treba da prikaže samo nekoliko narednih koraka, ne ceo put ka izlazu. U svakom trenutku treba omogućiti prelazak sa ekrana za igru na ekran sa glavnim menijem i obrnuto. Takođe, student bi trebalo da osmisli način za kretanje korisnika kroz lavirint uz pomoć tastera tastature (kada igrač unosi izbor, karakter izbora se ne sme videti, i ne mora se pritisnuti enter za prihvatanje znaka).

  - **2. student:** Implementira algoritam za generisanje nivoa i logiku igre
    Student bi trebalo da napravi generator lavirinta (nivoa). Prilikom započinjanja nove igre, bira se algoritam za generisanje. Student treba da realizuje bar dva takva algoritma. Kao uzor za generisanje lavirinta, student može da koristi izmenjeni DFS, Primov ili Kruskalov algoritam, po uzoru na projekat 7 – „Lavirint“. Student se ohrabruje da osmisli i implementira neki drugi algoritam za generisanje lavirinta umesto pomenutih. Ovaj student je zadužen i za implementaciju opcije za pomoć i rešavanje igre. Opcija za pomoć treba da prikaže samo nekoliko narednih igračevih koraka, ne celo rešenje.

- **3. student:** Implementira ponašanje računarskog protivnika (mumije)
  Računarskog protivnika (mumiju) bi trebalo implementirati na bar dva nivoa težine. Jednostavan računarski protivnik se trudi da uvek bude najbliži igraču, ima mogućnost da se zaglavi i ima mogućnost da ubije drugog računarskog protivnika (mumiju) ukoliko se nađu na istom mestu na mapi. Pametan računarski protivnik nema mogućnost zaglavljivanja i izbegava druge računarske protivnike (ne ubija ih). Za implementaciju pametnog računarskog protivnika se može koristiti neki od algoritama poput grananja i ograničavanja (eng. *branch & bound*), planinarenja (eng. *hill climbing*) i sl. Ovaj student je zadužen da osmisli i realizuje funkcije za precizno merenje vremena koje će se koristiti u programu. Ovaj student je zadužen i za održavanje liste najboljih skorova u igri, njeno učitavanje i snimanje u datoteku. Listu interno realizovati kao ulančanu listu, a tabelu najboljih skorova je potrebno na neki način zaštiti od neautorizovane promene van same igre.

Projektni zadatak 17: Igra „Yamb“

Autor: Tijana Kovačević
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Potrebno je napraviti program za igranje društvene igre Yamb za više igrača, od kojih neki mogu biti i kompjuterski igrači. Opšta pravila igre Yamb su data u prilogu. Na početku rada programa unosi se broj igrača, njihova imena i generiše se redosled kojim će igrati. Tokom igre, za igrača na potezu se ispisuje igrački listić koji sadrži tekući učinak. Igrač na potezu baca kockice i na osnovu dobijene kombinacije i sadržaja listića donosi odluku šta želi da odigra, koju unosi u formatu [*igra kolona*] (videti prilog). Na kraju igre, na ekranu se ispišu “listići” i skorovi svih igrača, kao i ime pobednika, a za svakog igrača pravi se fajl *ime_igraca.txt* gde se ispisuje njegov “listić”. Igru je moguće prekinuti, pri čemu se u fajl *prekid.txt* pamte “listići” i podaci o igračima, kako bi igra mogla da se nastavi. Napraviti jedan fajl po igraču (ime_igraca_dnevnik.txt) koji sadrži i listiće i kombinacije kockica u svakom potezu (dnevnik igre za svakog igrača). Realizovati kompjuterskog igrača koji se po pitanju igre ponaša ekvivalentno ljudskom. Omogućiti da tokom svakog poteza ljudski igrač zatraži pomoć kompjutera. Realizovati fajl *high_score.txt* u kom se čuvaju najvećih 10 postignutih rezultata, imena igrača koji su ih postigli i datum i vreme kada je to učinjeno.
  Igra je ograničena na osnovne 4 kolone – gde se rezultati bacanja upisuju u smeru *na gore, na dole, u proizvoljnom redosledu* i *Najava*.
- Raspodela aktivnosti:
  - **1. Student:** Realizuje kompjuterskog igrača i algoritam odlučivanja koji on koristi, kao i funkciju “pomoć kompjutera”
    Objašnjenje algoritma koji bi trebalo implementirati: za dobijenu kombinaciju od 5 kockica, na 32 načina možemo neke kockice zadržati, a neke zameniti. Listić nam daje informaciju o tome koja su nam polja trenutno na raspolaganju za popunjavanje. Za svaku od 32 potencijalne zamene, 1000 puta simuliramo bacanje i beležimo verovatnoću da se dobije svaka povoljna kombinacija (verovatnoća je broj dobijenih takvih kombinacija / 1000). Na kraju se, na osnovu ovih verovatnoća i hijerarhije igara koju treba dobro osmisliti, određuje koje zadržavanje kockica je satistički najpovoljnije. Na ovaj način kompjuterski igrač dolazi do odluke koje kockice zadržava, a koje menja. Kada dobije konačnu kombinaciju kockica (nema više menjanja), proverava skor za svako u datom trenutnku za upis dostupnih polja, i upisuje tako da skor bude najbolji u skladu sa hijerarhijom igara. Ovaj algoritam iskoristiti i za “pomoć kompjutera” koju može zatražiti ljudski igrač u svakom trenutku svog poteza.

  - **2. Student:** Realizuje interaktivni meni, rad sa ulazom i izlazom, struktura podataka
    Ovaj student efiniše strukturu podataka “listić” i strukturu podataka “igrač”, koja između ostalog sadrži pokazivač na funkciju koja realizuje igru. Ljudski igrač unosi odluku šta da igra i koje kockice da zadrži, a kompjuterski to automatski odlučuje, što predstavlja jedinu razliku u ponašanju. Interaktivni meni treba da sadrži sledeće opcije: nova igra, nastavak započete igre (ukoliko takva postoji), high-score lista, uključiti / isključiti pomoć kompjutera, kraj. Ovaj igrač pravi sve potrebne izlazne fajlove:
    ime_igraca.txt: fajl se formira na kraju igre i sadrži popunjen listić za datog igrača.
    ime_igraca_dnevnik.txt: fajl se formira na početku igre i u njega se “loguju” podaci o svakom potezu. Ispiše se prvo generisana kombinacija kockica, kockice posle dve zamene, i stanje na

listiću nakon odigranog poteza, kako bi mogao da se isprati svaki potez i po potrebi igra mogla da se rekonstruiše.
high_score.txt: fajl se formira pre nego što se program prvi put pokrene i eventualo se ažurira na kraju igre tako da sadrži 10 (ili manje, ukoliko nije odigrano dovoljno partija) najboljih skorova, imena igrača koji su ih postigli, datum i vreme kada su to učinili.
prekid.txt: fajl se formira u trenutku prekida tako da sadrži imena igrača po redosledu igranja, trenutno stanje na listićima i podatak ko je na potezu.
Omogućuje unos početnih podataka, dakle broj ljudskih igrača i njihova imena, broj kompjuterskih igrača. Tokom igre ispisuje stanje na listiću pre i nakon poteza svakog od igrača. Na kraju igre ispiše ime pobednika , listiće i skorove igrača. Ukoliko igrači odluče da nastave započetu igru, student realizuje čitanje potrebnih fajlova kako bi došao do podataka o stanju na listićima u trenutku prekida.
Korisnički interfejs za komunikaciju sa igračem i iscrtavanje Yamb listića mora biti dobro osmišljen i jednostavan. Iscrtavanje Yamb listića se uvek mora odvijati unutar istog ekrana, a izbor polja sa upis je potrebno intuitivno osmisliti, poželjno upotrebom kursorskih tastera.

- **3. Student:** Realizuje rad sa strukturama podataka "listić" i "igrač" (koje kreira drugi student), provere korektnosti, pomoćne funkcije, pomoćne strukture podataka, komunikaciju sa ljudskim igračem, generisanje "bacanja".
  Rad sa strukturama podataka – sve funkcije potrebne za maniplulaciju strukturama (alokacija, dealokacija, upis u polje, provera da li je popunjeno i slicno). Struktura "igrač" sadrži pokazivač na funkciju koja relaizuje igru. Ovaj student pravi funkciju za ljudskog igrača. Provere korektnosti – kada ljudski igrač unese odluku o potezu koji želi da odigra, potrebno je proveriti da li je potez korektan (možda to mesto već popunjeno, nije dostupno za upis, nemoguće odigrati tu igru sa datim kockicama ili je jednostavno nekorektan format unosa).
  Pomoćne funkcije – potrebno je proveriti koju sve igru može predstavljati data kombinacija kockica, zatim izračunati vrednost te igre, računati međurezultate i konačan rezulatat. Pomoćne strukture podataka – struktura koja prvom studentu dostavlja informaciju o poljma na koje se trenutno može izvršiti upis, struktura koja čuva podatke o hijerarhiji među igrama i slično. Komunikacija sa igračem u toku igre – unos odluke o potezu, eventualnom igranju najave, prekidu igre, potrebnoj pomoći kompjutera. U slučaju nekorektnog unosa, realizuje odgovarajuće akcije. "Bacanja" – generiše kombinaciju kockica i potrebne zamene za igrača na potezu.

Prilog: neka opšta pravila igre Yamb

Postoje 4 kolone:

\/ - upisuje se redom, odozgo na dole
/\ - upisuje se redom, odozdo na gore
/\/ - polja se popunjavaju proizvoljnim redosledom
N – najava – posle prvog bacanja igrac "najavi" kombinaciju koju igra, i posle dve zamene mora da upiše u to polje.

Tok igre:

U jednom potezu, predviđeno je najviše tri bacanja, a igrač nakon svakog može da upiše rezultat na listić ako smatra da ne može dobiti bolji rezultat (recimo, dobio je kentu iz prvog bacanja). Nakon upisa rezultata, sledeći igrač dolazi na red da igra.

"Baca se", odnosno u slučaju programa, generiše, početna kombinacija od 5 kockica. Na osnovu nje igrač odlučuje da li želi da igra *Najavu*. Ako želi, najavi određenu igru, recimo *triling*, i na kraju trećeg bacanja mora da upiše u odgovarajuće polje, ili da ga precrta (---), recimo u slučaju da nije dobio tri iste kockice.
Drugo i treće bacanje - igrač bira koje od 5 kockica želi da zadrži, a koje da zameni (baci ponovo), pritom može zameniti proizvoljan broj kockica, ili upisati rezultat u tabelu, ako smatra da ne može postići bolji rezultat.

U svim kolonama postoje sledeća polja - igre:

1. kategorija:
Brojevi 1 – 6: upisuje se zbir kockica iste vrste, recimo za 6 6 6 5 4, ako ovo odlučimo da tumačimo kao "šestice", upisaćemo 3*6 = 18.
Pravi se zbir polja po kolonama, ako zbir pređe 60 dodaje se bonus od 30 (obratiti pažnju na primer).

2. kategorija:
MAX – cilj je da zbir svih kockica bude što veći
MIN – cilj je da zbir svih kockica bude što manji
Ispod maksimuma i minimuma upisuje se rezultat ove kategorije, koji je (MAX – MIN)*jedinice (obratiti pažnju na primer).

3. kategorija:
Kenta – kentu čini 5 uzastopnih brojeva, 1 2 3 4 5 ili 2 3 4 5 6. Za kentu dobijenu iz prvog pokušaja upisuje se 66, iz drugog 56, a iz trećeg, 46.

Triling – čine ga tri iste kockice, recimo 6 6 6 5 4 je triling šestica. Upisuje se zbir te tri kockice + 30. Konkretno, 3*6 + 20 = 48.

Ful – čine ga "dve iste i tri iste". Recimo 6 6 6 5 5. Upisuje se zbir kockica + 30, dakle
3*6 + 2*5 + 30 = 58.

Kare – čine ga četiri iste kockice, recimo 5 5 5 5 4 je kare "petica". Piše se zbir te četiri cifre + 40, dakle 4*5 + 40 = 60.

Yamb – čine ga svih pet istih kockica, recimo 5 5 5 5 5, upisuje se njihov zbir + 50, dakle
5*5 + 50 = 75.

Pravi se zbir po poljima u ovoj kategoriji.
Na kraju se saberu zbirovi prve, druge i treće kategorije svih kolona, i to je konačan rezultat. Pobedjuje igrač koji ima najveći konačan skor.

Ukoliko se konačna kombinacija ne može upisati ni u jedno slobodno polje, piše se ---.

Podatke o igri koju želi da odigra igrač unosi u obliku *igra kolona*.
Igra predstavlja oznaku igre i to:

- broj od 1 – 6
- MAX, MIN
- kenta, triling, ful, kare, yamb
- --- (ako igrač želi da precrta neko polje)

Kolona predstavlja simbol kolone u koju zelimo da izvršimo upis, i to /\, \/ ili /\\/ .
Dakle, ako želimo da upišemo “šestice” u kolonu sa proizvoljnim smerom upisa, unećemo 6 /\\/. Ako želimo da upišemo triling u kolonu \/, unećemo triling \/.

Ako se igra najava, upis se vrši automatski.

Format listića, odnosno fajla ime_igraca.txt:

Ovako treba da izleda listić pri ispisu na ekran, i u svim ostalim izlaznim fajlovima.

kol.	\/	/\\/	/\	N	
1.	4	3	4	4	
2.	8	6	6	8	
3.	9	12	9	9	
4.	16	8	16	12	
5.	15	10	15	20	
6.	18	18	18	24	
Zbir:	100	57	98	107	362
MAX	28	27	28	29	
MIN	5	7	8	7	
Zbir:	92	60	100	88	340
Kenta	56	---	66	66	
Triling	38	38	35	38	
Ful	58	56	56	56	
Kare	64	60	60	64	
Yamb	75	80	---	---	
Zbir:	291	234	217	224	966
			Konacan rezultat:		1688

Projektni zadatak 18: Kartaška igra „Preferans“

Autor: Vladislav Guberinić
Rukovodilac projekta: Marko Mišić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Implementirati kartašku igru „Preferans“. Potrebno je implementirati *single player* režim igranja - igru igra samo jedan igrač, a poteze druga dva igrača odigrava računar korišćenjem veštačke inteligencije. Potrebno je implementirati konzolno grafičko okruženje (npr. korišćenjem biblioteke ncurses), veštačku inteligenciju za vršenje licitacije, odlučivanje o praćenju igre, kao i veštačku inteligenciju za samo odigravanje igre. Od studenata koji biraju ovaj projekat se očekuje određen nivo poznavanja ove igre. Prilikom izrade projekta, studenti mogu kao uzor koristiti sajt http://www.ipref.com/, kao i domaću implementaciju preferansa „Goran & Nidža“ koja se može pronaći na internetu. Prilikom izrade ovog projekta može biti korisna knjiga prof. Dragoša Cvetkovića, „Zanimljiva matematika – Preferans“.
- Raspodela aktivnosti:
  - **1. student**: Grafički interfejs, mešanje i deljenje karata, logovanje – Implementirati slučajno mešanje karata za fer igru. Implementirati i ručno podešavanje karata radi lakšeg testiranja. Program treba da podržava licitaciju, tablu za odigravanje karata, pisanje poena, mogućnost zvanja, davanja kontre...Potrebno je odabrati da se igra sa otvorenim ili zatvorenim kartama (radi boljeg uvida u kvalitet računarskog protivnika). Ovaj student implementira ispis svih podataka vezanih za igru. Primeri podataka su: broj potencijalnih nošenja kad pratimo ili broj očekivanih nošenja pre licitacije. Ovime se stiče uvid u igru i kvalitet veštačke inteligencije. Potrebno je omogućiti rekonstrukciju igre u grafičkom okruženju na osnovu ovih podataka. Takođe, treba podržati snimanje i učitavanje započete igre.
  - **2. student**: Licitacija, zamena karata, praćenje – Ovaj igrač implementira osnovne algoritme vezane za obavljanje licitacije, zamene karata i donošenja odluke o praćenju. Računarski protivnik treba da zna da oceni koliko ima nošenja u ruci. Treba da zna da bela nosi, da su makaze jedno sigurno potencijalna dva, da solo pop nije nošenje, itd. Treba da zna da je licitranje sa manje od četiri aduta ili bez keca u adutu vrlo sumnjivo. Da prepoznaje uvr betl i betl sa solo osmicom i slične stvari, kao i igru sans. Takođe, ovaj igrač treba da odredi odgovarajuće procene. Procene da li je bolje imati dugu boju ili neku kraću a jaču. Procene na osnovu toga šta su igrači licitirali, npr. sans bez boje je ok, samo ako ni jedan igrač nije licitirao do boje do koje smo mi stali. Implementirati odlučivanje o tome da li pratiti. Ovde voditi računa o tome šta je kupljeno (u slučaju licitacije), dokle je ko od igrača licitirao, ko ima polaz, ko je već krenuo. Računanje potencijalnog štiha (solo as, bela...), malo verovatnog štiha (dupli pop ako mi polazimo, treća dama...) i sigurnih kombinacija (npr AKQJ u boji ili peta dama).
  - **3. student**: Odigravanje partije - Polaz u zavisnosti od karata, da li smo levim desni ili mi vodimo. Isterivanje i brojenje aduta i nalaženje (duple) seče. Pravila da se kroz svog kreće velikom i kroz protivnika malom (očigledno ne ako imamo veliku u toj boji). Kretanje iz boje saigrača. Odigravanje betla i sansa. Uzimanje u obzir kupljenih karata. Prebrojavanje preostalih karata. Ovaj student implementira demonstracioni mod igre u kome su sva tri igrača simulirana, odnosno gde računar igra protiv samog sebe.

Projektni zadatak 19: „Vertical scrolling shooter 1941“ igra

Autor: Stefan Pantelić
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Realizovati verziju arkadnih vertical shoot’em up igara iz serije 1941 http://en.wikipedia.org/wiki/1941:_Counter_Attack. Potrebno je implementirati grafiku u konzolnom okruženju, mehaniku igre i ponašanje računarski vođenih protivnika. Igre iz serije 1941 su vertical shooter igre kod kojih se napredak i akcija u igri (nivou) odvija vertikalnim kretanjem po ekranu. Cilj igre je uništavanje određenog broja protivnika, nakon čega na kraju svakog nivoa dolazi do sukoba sa glavnim protivnikom (“boss-om”). Igrač tokom igre dobija poene za svakog uništenog protivnika, a povremeno unipteni protivnik ostavlja za sobom određene bonuse (pojačanja) kao što su dodatni život, povećanje snage, pojačavanje projektila, dodatne bombe, specijalna municija i slično. Svaki uništeni protivnik donosi igraču određenu količinu municije, može naneti definisanu štete i donosi određen broj poena nakon uništenja. Potrebno je realizovati nekoliko tipova protivnika, po uzoru na originalnu igru.
- Raspodela aktivnosti:
  - **1. student**: Grafički interfejs, mešanje i deljenje karata, logovanje – Ovaj student kreira konzolni grafički korisnički interfejs igre. Potrebno je napraviti interaktivni meni u kojem korisnik pokreće igru, učitava i snima započetu igru, pregleda listu najboljih skorova i vrši podešavanja igre. Takođe, potrebno je obezbediti i opciju koja sadrži kratko objašnjenje igre i dostupnih kontrola. Potrebno je realizovati i modele aviona i protivnika, realizovati efekte i definisati funkcije koje će iscrtavati sve grafičke elemente. Pored toga, prvi student realizuje listu najboljih skorova koja bi trebala da bude zaštićena od promena van igre.
  - **2. student**: Mehanika igre – Ovaj student kreira mehaniku igre. Dohvata kontrole igrača, položaje i akcije botova kroz funkcije koje implementira treći student, prati stanje igre, a zatim preko grafičkih funkcija vrši prikazuje stanje igre u konzoli. Drugi student je odgovoran za generisanje nivoa, što obuhvata broj i vrstu botova, vrstu boss-a , broj slobodnih pojačanja (power-up-ova i sl). Svi navedeni parametri zavise od izabrane težine igre. Drugi student implementira detekciju kolizije i pogodak, realizuje sistem rezultata i njegovog izračunavanja, kao i efekte pojačanja.
  - **3. student**: Veštačka inteligencija i računarski protivnik – Student realizuje ponašanje računarskih protivnika (botova). Potrebno je realizovati najmanje tri različite vrste protivnika i najmanje jednog “bosa”. Botovi se ponašaju u skladu sa izabranom težinom igre. Treći student piše funkcije koje kao parametre primaju stanje igre i zatim „odlučuju“ o odgovarajućoj akciji. Na botove ne utiču pojačanja, ali je moguće da ih pokupe kako bi onemogućili igrača u njihovom sakupljanju. Takođe, treći student realizuje demonstracioni režim igre u kome se i igračev avion kontroliše od strane računara i koji treba da pokaže ispravnost veštačke inteligencije.

Projektni zadatak 20: Slagalica “Game of Fifteen”

Autori: Slobodan Stojanović
Rukovodilac projekta:
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: igra se sastoji od 15 pločica (http://en.wikipedia.org/wiki/15_puzzle), koje su obeležene od 1 do 15 stavljenih u „kutiju“ dimenzija 4x4. Jedna od 16 pozicija je prazna i u nju mozemo pomerati susedne pločice, sto dalje izaziva pojavu novog praznog mesta.
  Ideja je da počevši od slučajno izabrane konfiguracije, pomeranjem pločica na praznu poziciju dođemo do ciljne konfiguracije (vidi sliku). Igra se može igrati i sa smanjenom tablom dimenzija 3x3 sa 8 pločica. Cilj igrača je da do ciljne konfiguracije dođe u što manjem broju koraka.
- Raspodela aktivnosti:
  - **1. student**: Grafički interfejs i režije
    Zadatak ovog studenta je da omogući komunikaciju između računara i korisnika u grafičkom konzolnom okruženju. Potrebno je realizovati grafiku igre, osmisliti način na koji će korisnik pomerati pločice prilikom igranja, zadavanje početnog stanja igre i omogućavanje automatskog rešavanja igre pomoću funkcija koje realizuju drugi i treći student. Potrebno je i omogućiti pomoć igraču u vidu sugerisanja narednog koraka. Takođe, ovaj student je zadužen za integraciju funkcija koje realizuju drugi i treći student u jednu celinu. Potrebno je omogućiti i učitavanje i snimanje započete partije, podešavanje boja u igri i prikaz liste najboljih skorova, kao i druga podešavanja igre.
  - **2. student:** Implemetancija algoritma za rešavanje 8-puzzle igre
    Ovaj student implementira rešavanje ove igre u smanjenom izdanju (8 pločica, kutija 3x3) upotrebom A* algoritam. Takođe, ovaj student implementira heuristiku za izračunavanje cene puta između čvorova korišćenjem bar dve dostupne heuristike. Zbog potreba algoritma, neophodno je efikasno implementirati prioritetni red. Zbog upoređivanja performansi, prioritetni red je potrebno implementirati bar na dva načina – jedan u vremenu O(n), a drugi u vremenu O(log n) korišćenjem binarnog *heap* stabla. Varijanta prioritetnog reda i heuristike koja se koristi treba da se bira u podešavanjima igre.
  - **3. student**: Implementacija algoritma za rešavanje 15-puzzle igre
    Treći student je zadužen za realizaciju structure podataka kojom se definiše sama igra, kao i za implementaciju funkcija za pristup toj strukturi. S obzirom da A* algoritam troši previše memorije za rešavanje igre sa 15 pločica, ovaj student implementira IDA* algoritam koji predstavlja varijaciju A* algoritma i omogućava rešavanje igre u potpunom izdanju (15 pločica, kutija 4x4). Takođe, ovaj student realizuje listu najboljih skorova sa zaštitom od izmene van igre.

PRILOG: rešivost početne konfiguracija, prioritetni red, heuristike

Rešivost:
Važno je napomenuti da nisu sve konfiguracije igre rešive, (poznato je da polovina njih nije). Sve neparne permutacije pločica su nerešive (http://en.wikipedia.org/wiki/Parity_of_a_permutation), dok su parne permutacije rešive. Pre nego što se krene u rešavanje bilo bi potrebno na neki način odrediti da li je početna konfiguracija rešiva ili nije. Naivan način je da se pokrene algoritam za datu konfiguraciju i za konfiguraciju gde su zamenjene dve susedne pločiče u nekom redu. Ukoliko postoji rešenje za konfiguraciju u kojoj su zamenjene pločice pre nego što se to dogodi za konfiguraciju koja se rešava, u pitanju je neparna permutacija. Takođe, rešivost se može ustanoviti i analizom rasporeda pločica, što se ostavlja studentima da pogledaju u otvorenoj literaturi.

Prioritetni red:
Implementacija prioritetnog reda čija je vremenska složenost operacija PQ_INSERT i PQ_MIN_DELETE reda O(n) je neprihvatljiva sa stanovišta praktične upotrebe. Stoga se preporučuje implementacija prioritetnog reda pomoću binarnog *heap* stabla. Binarno *heap* stablo je skoro kompletno stablo koje se efikasno implementira pomoću sekvencijalne (vektorske) implementacije. Najprioritetniji element se nalazi u korenu stabla. Složenost dohvatanja najprioritetnijeg elementa je O(1), a ažuriranje stabla se vrši u vremenu O(log n), bilo nakon dodavanja novog elementa ili dohvatanja postojećeg. Više informacija postoji u otvorenoj literaturi, kao i knjizi prof. Mila Tomaševića u delu o *Heapsort* algoritmu.

A* algoritam:
A* predstavlja "best first search" algoritam koji nalazi put u grafu između početnog čvora i ciljnog (goal) čvora. U našem slučaju početni čvor će biti početno stanje igre a ciljni čvor predstavlja završnu konfiguraciju (konfiguraciju gde su sve pločice na svom mestu). Ovaj algoritam koristi heuristiku za izračunavanje cene puta između čvorova i na taj način uvek bira čvor sa najmanjom cenom (best) i njega prvo posećuje. Funcija koja računa cenu puta iz čvora u kome se trenutno nalazimo do sledećeg se sastoji od a) cene puta do čvora u kome se nalazimo b) cene puta do novog čvora koju računamo koristeći neku heuristiku. Zbir a) i b) daje cenu puta od početnog do krajnjeg čvora. Za dobre performanse ovog algoritma od velike je važnosti dobra implemetacija prioretetnog reda!

IDA* algoritam:
IDA* algoritam je varijacija A* algoritma koja ima znatno manju prostornu složenost. Algoritam ne koristi prioritetni red, već modifikovani DFS algoritam. Algoritam koristi određenu heuristiku da sračuna maksimalnu cenu koja sme da se upotrebi u svakoj iteraciji algoritma. Kada algoritam poseti čvor čija je cena veća od maksimalne cene za datu iteraciju, on zaključuje da taj put sigurno ne vodi do rešenja i radi odsecanje (tzv. *cutoff*) puta i vraća se nazad (*backtrack*), gde isprobava druge puteva koji ga mogu dovesti do rešenja. Maksimalna cena za svaku iteraciju se računa počevši od prve. U prvoj iteraciji se cena, računa nekom heuristikom za početni čvor. U svakoj sledećoj iteraciji maksimalna cena se računa kao minimalna cena čvorova koju su bili odbačeni (odnosno nisu izabrani pri DFS pretrazi). Cena puta od početnog čvora do nekog čvora se računa kao i u slučaju A* algoritma.

Heuristika:
U ovom slučaju potrebno je koristiti neku heuristiku da se nađe cena odgovarajuće konfiguracija. Jedna takva heuristika je tzv. Manhattan distanca. Manhattan distanca je razdaljina (distance) između dve tačke merene po osama pod pravim uglom. Na primer, neka su $X_i(s)$ i $Y_i(s)$ x i y kordinate i-te pločice u konfiguraciji s a neka su nadvučeno X_i i Y_i kordinate i-te pločice u ciljnoj konfiguraciji. Manhattan cena takve konfiguracije ce biti (u slučaju 8-puzzle):

$$h(s) = \sum_{i=1}^{8} (|x_i(s) - \overline{x}_i| + |y_i(s) - \overline{y}_i|).$$

Projektni zadatak 21: Igra „Crazy Snake“

Autor: Vidor Gencel, Ivan Dimitrov
Rukovodilac projekta:
Broj studenata: 3

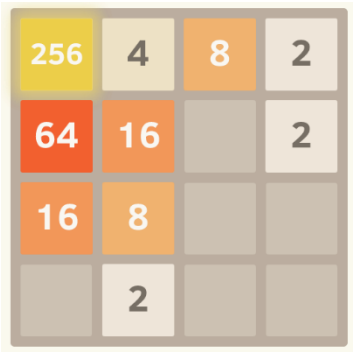
Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Igra „Crazy Snake“ je bazirana na klasičnoj «Snake Game» igri. Cilj igre je skupiti što više tokena za određeni vremenski period. Zmija se kontroliše strelicama na tastaturi. Na mapi, osim slučajno postavljenih tokena, nalaze se i prepreke u vidu zidova. Ukoliko zmija udari u zid igra se završava. Treba obezbediti da ukoliko zmija izađe van okvira mape da nastavi da se kreće sa suprotne strane. Nakon što zmija glavom pokupi token njena dužina se povećava. U svakom trenutku na mapi se nalazi samo jedan token. Ukoliko prođe vremenski period, a igra se ne završi prethodno udarcem u zid, igraču se prikaže statistika prikupljenih tokena i broj tokena koji je mogao da pokupi krećući se, uvek, najkraćim putem. Ukoliko je rezultat u najboljih 10, traži se unos imena i igrač se upisuje u listu koja se čuva u datoteku koja treba da bude zaštićena od izmena van programa..
- Raspodela aktivnosti:
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika
    Glavni program sadrži interaktivni meni sa opcijama nova igra, opcije, pomoć, o igri, najbolji skorovi, autori i izlazak. U opcijama se podešava vreme koliko traje partija, veličina mape, brzina zmije (težina igre), brisanje tabele najboljih skorova i sl. Kada se odabere nova igra, iscrtava se mapa sa postavljenim zidovima (drugi student). Izgled mape i zidova podrazumeva korišćenje boja i jednostavne ASCII grafike. U svakom trenutku treba omogućiti prelazak sa ekrana za igru na ekran sa glavnim menijem i obrnuto. Student bi trebalo da osmisli način za pomeranje zmije uz pomoć tastera tastature (kada igrač unosi izbor, karakter izbora se ne sme videti, i ne mora se pritisnuti enter za prihvatanje znaka). Takođe, potrebno je implementirati učitavanje i snimanje već započete igre.
  - **2. student:** Implementira logiku igre
    Ovaj student implementira odgovarajuću strukturu podataka za smeštanje mape i elemenata na mapi, kao i algoritme za raspoređivanje (generisanje) zidova i tokena po mapi. Student realizuje i sistem za praćenje poena. Na početku je skor nula, a nakon skupljanja tokena broj poena se povećava, kao i dužina zmije. Potrebno je i implementirati algoritam koji ažurira stanje mape posle svakog koraka zmije.
  - **3. student:** Implementira algoritam za određivanje najkraćeg puta
    Zadatak studenta je da implementira algoritam za određivanje najkraćeg puta između zmije i tokena. Pomoću ovog algoritma student određuje najmanji broj koraka (vreme) koji je potreban zmiji da dođe do zadatog tokena. Pomoću ovog algoritma moguće je izračunati maksimalan skor u jednoj igri. Po završetku igre, ukoliko je ostvaren rezultat koji je u 10 najboljih, igraču se traži da unese ime i rezultat se čuva u datoteku što implementira treći student. Datoteka bi trebalo da bude tekstualnog tipa, uz zaštitu od neovlašćene promene van igre. Takođe, ovaj student realizuje demonstracioni režim igre, gde zmiju kontroliše računar do isteka zadatog vremena.

Projektni zadatak 22: Igra „2048“

Autor: Ivan Dimitrov, Vidor Gencel
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		



- Kratak opis projekta: Igra „2048“ se igra na kvadratnoj tabli dimenzija 4x4. Polje table može da bude prazno ili da sadrži pločicu određene vrednosti. Na svakoj pločici je ispisana njena vrednost. Na početku igre na tabli se nalazi jedna pločica sa vrednošću 2 na slučajno generisanoj poziciji. Pritiskom na taster neke od strelica na tastaturi sve pločice na tabli se pomeraju u odgovarajućem smeru, i ukoliko se jedna pločica pomera ka drugoj, a obe imaju istu vrednost, tada se one spoje i formiraju pločicu koja ima vrednost zbira dve pločice koje su se spojile. U jednom koraku može da se desi i više spajanja. Posledica ovog pravila jeste da pločice mogu da imaju vrednost koja je samo stepen broja dva. Nakon pomeranja pločica na tablu se dodaje nova pločica sa vrednošću 2 ili 4 (verovatnoće pojavljivanja: 0.8, 0.2) na slučajno odabranom mestu. Cilj igre je da se napravi pločica sa vrednošću **2048**. Igraču je na raspolaganju «hint» dugme koje će automatski odigrati optimalan potez. Kraj igre je ukoliko se naprvi pločica zadate vrednosti ili kada nema više raspoloživih poteza, a nije napravljena pobednička pločica. Tada se igra završava i upisuje se rezultat u datoteku
- Raspodela aktivnosti:
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika
    Glavni program sadrži interaktivni meni sa opcijama nova igra, opcije, pomoć, o igri, najbolji skorovi, autori i izlazak. U opcijama se podešava veličina table, vrednost pločice koja završava igru, brisanje tabele najboljih skorova i sl. Kada se odabere nova igra, iscrtava se tabla definisanih dimenzija, a zatim se automatski postavlja prva pločica. Izgled table i pločica podrazumeva korišćenje boja i karakterne ASCII grafike. Pri pozivanju pomoći program treba automatski da odigra optimalan potez. U svakom trenutku treba omogućiti prelazak sa ekrana za igru na ekran sa glavnim menijem i obrnuto. Takođe, student bi trebalo da osmisli način za pomeranje pločica uz pomoć tastera tastature.
  - **2. student:** Implementira logiku igre
    Student bi trebao da implementira odgovarajuću strukturu podataka za smeštanje table i pločica, kao i algoritam koji ažurira stanje table posle odigranog poteza (prikaz promene radi prvi student). Potrebno je realizovati sistem za praćenje poena. Na početku je skor nula. Kada se spoje dve pločice iste vrednosti (npr. dve dvojke) na skor se dodaje zbir spojenih pločica.
  - **3. student:** Implementira algoritam za «hint»
    Pošto se igra u velikoj meri oslanja na slučajnost, student ne treba da realizuje algoritam koji pronalazi kompletno rešenje (jer nije moguće), već treba da odredi koji je sledeći najbolji potez. Na izbor sledećeg najboljeg poteza utiču sledeći faktori: potez kod koga se najviše pločica spaja, potez koji će spojiti pločice najveće vrednosti, potez koji vodi računa o praznim poljima gde će se nakon pomeranja generisati nova pločica. Student mora da nađe kompromis između navedenih faktora u realizaciji algoritma. Po završetku igre, ukoliko je ostvaren rezultat koji je u 10 najboljih, igraču se traži da unese ime i rezultat se čuva u datoteci koja treba da bude zaštićena od izmena van programa.

Projektni zadatak 23: Jednostavni menadžer datoteka

Autor: Milan Marinković, Marko Mišić
Rukovodilac projekta: ?
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Potrebno je realizovati aplikaciju koja implementira određene funkcionalnosti menadžera datoteka, poput Total Commander-a. Alat treba da iščitava sadržaj fajl sistema (koristeći usluge operativnog sistema ili neke biblioteke) i omogući određene manipulacije nad fajlovima. Potrebno je omogućiti pronalaženje i uklanjanje duplikata fajlova na osnovu kriterijuma koje zadaje korisnik, sinhronizaciju dva foldera i višestruko preimenovanje datoteka korišćenje džoker znaka (* i ?). Fajlovi se mogu porediti po imenu, ekstenziji, vremenu kreiranja i modifikovanja, ili putem sadržaja bit po bit. Program treba da posebno podrži određene popularne formate fajlova (poput BMP, JPEG, MP3, WAV, ZIP i slično) kod kojih bi se omogućilo dubinsko poređenje na osnovu karakteristika formata. Korisnik treba da ima mogućnost selekcije fajlova pomoću džoker znakova i višestruko preimenovanje (*multi-rename* opcija). Korišćenjem funkcija za poređenje, potrebno je omogućiti sinhronizaciju dva direktorijuma (po zadatim kriterijumima). Takođe, alat treba da omogući rad iz komandne linije. Za sve pretrage i akcije, alata treba da omogući kreiranje tabelarnog izveštaja koji se može snimiti u CSV formatu. Pre obavljanja bilo kakvih destruktivnih akcija, alat mora korisniku da zatraži odobrenje.
- Raspodela aktivnosti
  - **1. student**: Grafički interfejs i rad sa greškama
    Treba predvideti dva načina rada: pomoću komandne linije i putem interaktivnog menija. U interaktivnom načinu rada se prikazuju meniji sa mogućim operacijama za manipulaciju datotekama, dok se u režimu rada u komandnoj liniji svi potrebni parametri i putanje prosleđuju programu. Grafički interfejs ove aplikacije treba da sadrži okvir u kome će korisnik moći da pregleda strukturu direktorijuma po uzoru na *shell* Total Commander-a. Student samostalno definiše odgovarajući format za parametre prilikom rada iz komandne linije u zavisnosti od podržanih opcija programa. Ukoliko se neki parametar ne navede uzima se default vrednost koja je zapisana u konfiguracionom fajlu programa ako taj fajl postoji, odnosno default vrednosti koje su fiksirane u programu u suprotnom. Takođe, treba voditi računa o međusobnoj isključivosti određenih parametara i upozoriti korisnika ako do toga dođe. Treba omogućiti posebnu opciju u meniju za kreiranje i izmenu parametara konfiguracionog fajla. Takođe, ovaj student je zadužen za generisanje izveštaja. Poseban parametar –l treba da omogući pravljenje programskog dnevnika (*log*). U dnevniku se beleže svi karakteristični koraci prilikom rada programa i može se smatrati proširenim izveštajem.
  - **2. student**: Rad sa fajl sistemom, manipulacija formatima i poređenje
    Zadatak ovog studenta je da ostalima pruži funkcije koje se tiču rada sa fajl sistemom kao što su dohvatanje svih datoteka jednog direktorijuma, otvaranje i zatvaranje fajlova, brisanje fajlova. Drugi skup funkcija se odnosi na rad sa podržanim formatima i čitanjem njihovih zaglavlja kako bi se omogućila dubinska pretraga i poređenje fajlova. Takođe, ovaj student je zadužen za implementiranje bilo koji internih struktura koje će se koristiti za komunikaciju između interfejsa i biblioteke (zadaci studenta 1. i 3. respektivno). Treći skup funkcija koje realizuje ovaj student se odnosi na selekciju fajlova pomoću džoker znakova i implementiranje *multi-rename* opcije.

- **3. student:** Poređenje fajlova, sinhronizacija direktorijuma
  Ovaj student treba da osmisli i realizuje poređenje fajlova, nalaženje duplikata i sinhronizaciju direktorijuma na osnovu podataka koje mu dostavi drugi student. Potrebno je osmisliti način inteligentnog poređenja datoteka, kao i korišćenja podataka prilikom dubinskog poređenja specifičnih formata. Prilikom sinhornizacije direktorijuma, korisniku treba omogućiti sinhronizaciju sleva na desno, sdesna na levo i u oba smera, ignorisanje datuma datoteka i sl. Na ovom studentu je da pronađe algoritme i strukture podataka koji najviše odgovaraju problemima poređenja, pretrage i sinhronizacije i da ih implementira radi njihovog rešavanja.

Projektni zadatak 24: Igra „Worms“

Autor: Bogdan Bebić
Rukovodilac projekta: Bogdan Bebić
Broj studenata: 3

Student	Broj indeksa	e-mail
1.		
2.		
3.		

- Kratak opis projekta: Potrebno je realizovati uprošćenu verziju popularne igre Worms. Igra će biti namenjena za jednog (singleplayer) ili dva igrača (multiplayer). Potrebno je implementirati grafički prikaz, logiku (*engine*) same igre, generisanje nivoa i veštačku inteligenciju protivnika. U okviru igre, igrač gađa protivnika projektilom skidajući mu *health* poene kada projektil padne dovoljno blizu protivnika. Cilj igre je da igrač skine sve *health* poene protivnika pre nego što izgubi svoje *health* poene.
- Raspodela aktivnosti
  - **1. student**: Glavni program, meni za interaktivan rad, jednostavna grafika

    Zadatak ovog studenta je realizacija svih grafičkih elemenata igre. Student treba da napravi interaktivni meni u kome se može pokrenuti nova igra, nastaviti igra, mogu podesiti parametri same igre (težina protivnika, veličina mape, odabir single/multiplayer partija i sl) i pogledati lista najboljih skorova. Potrebno je obezbediti sve funkcije koje će se koristiti pri iscrtavanju samog nivoa, a koje će pozivati 2. student.
  - **2. student:** Implementira logiku igre

    Ovaj student implementira odgovarajuću strukturu podataka za generisanje i smeštanje mape i elemenata na mapi. Zadatak ovog studenta je da dohvata komande igrača, komande protivnika (koje obezbeđuje 3. student) i da ažurira stanje i pozove funkcije 1. studenta radi iscrtavanja novog stanja na konzoli. Logika igre obuhvata i analizu trenutnog stanja mape (promene na mapi, vođenje računa o trenutnom skoru i sl). Takođe, ovaj student je zadužen za implementaciju nastavljanja igre na osnovu funkcija koje implementiraju druga dva studenta.
  - **3. student:** Implementira računarskog protivnika

    Zadatak studenta je da implemetira algoritme koji upravljaju računarskim protivnikom, kao i samim igračem za demonstracioni mod igre. Potrebno je napisati funkcije koje odlučuju o akcijama protivnika u narednom koraku (kretanje u jednom od dva smera ili mirovanje, lansiranje projektila) na osnovu trenutnog stanja na mapi i njihovog odabira (različite težine protivnika se biraju u meniju). Potrebno je napraviti bar tri težine sa različitim algoritmima za igranje. Listu najboljih skorova bi trebalo zaštiti od neovlašćene promene van igre.