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

Predmet: Praktikum iz programiranja 2 (SI1PP2)
Predmetni asistent: Marko Mišić
Školska godina: 2009/2010.

Projektni zadaci za samostalni rad

Verzija dokumenta: 1.2

Uvod

Ispit SI1PP2 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 SI1PP2. Samim tim postoji problem određivanja primerene težine zadataka, tako da lošijim 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 dobiju preko 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 SI1PP2, 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 predmetnog asistenta (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 predemtnog asistenta 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, ohrabruje se upotreba odgovarajućih SVN (Subversion) ili CVS (Concurrent Versioning System) alata za praćenje različitih verzija programa koje studenti mogu pronaći na internetu. 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 demonstratorom.

Svaki tim redovno prati i nadgleda zaduženi predmetni demonstrator i predmetni asistent. 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

asistentom 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, asistent.

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 asistenta 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:

Autor: Marko Mišić (marko.misic@etf.rs)
Rukovodilac projekta: Đorđe Mijović (?)/Nemanja Đorđević(?)/Marko Mišić
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 2:

Autor: Grupa studenata / Đurđević Đorđe (zorz@etf.bg.ac.rs) / Marko Mišić (marko.misic@etf.rs)
Rukovodilac projekta: Marko Mišić
Broj studenata: 3

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

- Kratak opis projekta: Kompresija podataka korišćenjem DEFLATE algoritma. 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 i DEFLATE algoritam
    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. Ovaj student realizuje DEFLATE algoritam uz pomoć gotovih struktura koje realizuju studenti 2 i 3. 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. 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**: Dinamički i statički Huffman-ov alagoritam
    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 studentima 1 i 3. Ovaj student treba da implementira i Burrows-Wheeler-ovu transformaciju, koja se koristi kao međukorak u kompresiji koji povećava stepen kompresije Ova transformacija se se opciono koristi u programu. Takođe, potrebno je omogućiti parametrizaciju kod binarnih stabala tako da korisnik može da zada način kodiranja leve, odnosno desne grane u binarnom stablu (standardno se leva strana kodira bitom vrednosti 0).
  - **3. student:** LZW algoritam i rad sa datotekama
    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.

Projektni zadatak 3:

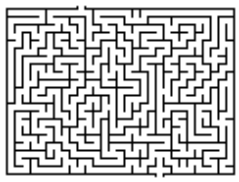
Autor: Đurđević Đorđe (zorz@etf.bg.ac.rs) / Marko Mišić (marko.misic@etf.rs)
Rukovodilac projekta: (?)
Broj studenata: 3

- 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**:
    - 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 4:

Autor: Mirko Francuski (mire.etf@gmail.com)
Rukovodilac projekta: Mirko Francuski (?) / Djordje Maksimović (?) (djordjem88@gmail.com)
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. 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.

- **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 i za održavanje liste najboljih skorova u igri, njeno učitavanje i snimanje u datoteku. Listu interno realizovati kao ulančanu listu.

Projektni zadatak 5:

Autor: Mirko Francuski (mire.etf@gmail.com)
Rukovodilac projekta: Mirko Francuski (?)
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. Hala slavnih sadrži imena i rezultate (broj gađanja) za 10 najboljih igrača i implementira se kao ulančana lista.

- **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.

Projektni zadatak 6:

Autor: Đurđević Đorđe (zorz@etf.bg.ac.rs)
Rukovodilac projekta: Vladimir Mihajlovski (?) (vladamih@gmail.com)
Broj studenata: 3

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

- Kratak opis projekta: pravljenje softvera ja jednostavnu interaktivnu obradu digitalnog zapisa zvuka iz WAV fajlova, putem jednostavnog menija. Editovanje uključuje pojačanje i utišavanje zvuka, potiskivanje šuma, promenu učestanosti reprodukcije, stvaranje efekata, mešanje sadržaja dva WAV fajla. Sve aktivnosti se beleže u dnevniku rada, ako je ova opcija aktivna. Stavke menija su:
    1. učitavanje WAV fajla
    2. snimanje WAV fajla
    3. zatvaranje WAV fajla (brisanje iz memorije)
    4. stvaranje novog WAV fajla (u memoriji)
    5. unarne operacije nad WAV fajlom
    6. binarne operacije nad WAV fajlovima
    7. uključivanje/isključivanje mogućnosti pravljenja dnevnika rada (log fajl)
    8. izlaz
- Raspodela aktivnosti
    - **1 student**: realizacija glavnog programa koji omogućava korisniku interaktivan rad putem jednostavnog menija. Na glavnom meniju se u svakom trenutku ispisuju imena svih učitanih WAV fajlova (može ih biti više). Ovi fajlovi ne mogu da se snimaju. U nastavku se ispisuje spisak svih novokreiranih WAV fajlova (ili fajlova dobijenih obradom) koji mogu da se snimaju. Student realizuje stavke 1, 2 3, 6, 7 i 8 iz gornjeg spiska stavki menija. Student treba sam da potraži format WAV fajla na Internetu i da ga prouči. Student realizuje sledeće binarne operacije:
        1. nadovezivanje drugog fajla na kraj prvog.
        2. umetanje drugog fajla u prvi uz zadavanje pozicije gde se vrši umetanje
        3. mešanje drugog fajla sa prvim od zadate pozicije i sa zadatim koeficijentom. Koeficijent mešanja može biti fiksan (čita se sa standardnog ulaza) ili se čita niz koeficijenata iz tekstualnog fajla, a svaki se primenjuje na jedan odbirak.
        4. nadovezivanje drugog fajla na kraj prvog uz efekat prelaza (prvi se postepeno utišava, drugi se postepeno pojačava). Vremensko trajanje prelaza se unosi preko standardnog ulaza.
        5. razlika sadržaja fajlova: prvi – drugi
        6. množenje sadržaja fajlova: prvi * drugi

      Napomena: Ako učestanost odbiraka nije ista za oba fajla, onda se automatski učestanost drugog prilagođava učestanosti prvog.

    - **2 student**: stvaranje novog WAV fajla. Novi WAV fajl se stvara tako što se od korisnika dobiju karakteristične informacije fajlu: učestanost odbiraka i trajanje. Tako kreiran WAV fajl može da se popuni sledećim informacionim sadržajem:
        1. prazan fajl (ceo informacioni deo popunjen nulama)
        2. uzlazna rampa (uz zadavanje amplitude i periode)
        3. silazna rampa (uz zadavanje amplitude i periode)
        4. trougaoni (testerast) signal (uz zadavanje amplitude i periode)

5. četvrtast signal (uz zadavanje amplitude i periode)
6. sinusoidni signal (uz zadavanje amplitude i periode)
7. učitavanje oblika signala iz tekstualnog fajla (uz mogućnost ponavljanja učitanog signala ako je kraći od ukupnog trajanja kreiranog WAV fajla)

- **3 student**: realizacija sledećih unarnih operacija nad WAV fajlom:
  1. pojačavanje
  2. utišavanje
  3. potiskivanje šuma
  4. stvaranje eho-efekta
  5. promena učestanosti reprodukcije
  6. invertovanje vremenske ose signala

  Za tačke 1-3 korisnik može da bira vremenski i amplitudni opseg u okviru kojeg se primenjuje data operacija. Za tačku 4, korisnik može da unese eho-parametre preko standardnog ulaza, da ih pročita iz tekstualnog fajla ili da primeni neki unapred definisan.

Projektni zadatak 7:

Autor: Branko Kokanović (branko.kokanovic@gmail.com)
Rukovodilac projekta: Ivan Kuraj (?) (ivcha@šietf.org)
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 8:

Autor: Tijana Kovačević (tijana.kovachevic@gmail.com)
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