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: 2011/2012.

Projektni zadaci za samostalni rad

Verzija dokumenta: 1.4

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: Bomberman

Autor: Miloš Tijanić
Rukovodilac projekta: Miloš Tijanić (sherincall@gmail.com)
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.

- 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.

  - **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.

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 2: Muzička biblioteka

Autor: Miloš Tijanić / Dušan Jovanović
Rukovodilac projekta: Miloš Tijanić (sherincall@gmail.com) / Dušan Jovanović (dusan016@gmail.com)
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 3: *Game of Life* simulacija

Autor: Filip Živković (filip.zivkovic@sietf.org)
Rukovodilac projekta: Filip Živković (filip.zivkovic@sietf.org)
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 4: Alat za kompresiju i dekompresiju podataka

Autor: Grupa studenata / Đurđević Đorđe (zorz@etf.bg.ac.rs) / Marko Mišić (marko.misic@etf.rs)
Rukovodilac projekta: Filip Živković (?) / Dušan Jovanović (?) / Miloš Tijanić (?) / Marko Mišić
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 5: DivX titl konvertor

Autor: Marko Mišić (marko.misic@etf.rs)
Rukovodilac projekta: 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 6: Generator HTML galerija

Autor: Đurđević Đorđe (zorz@etf.bg.ac.rs) / Marko Mišić (marko.misic@etf.rs)
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.

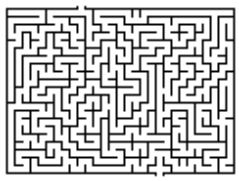
    - **2 student**: realizuje tačku 3.3 i deo tačke 7. U okviru tačke 3.3, student treba da čita sadržaj zadatog direktorijuma, pronađe imena svih fajlova koji odgovaraju zadatom imenu sa džoker znakom i doda ih u spisak slika. U okviru tačke 7, student treba da napravi kostur HTML stranice, na osnovu definisanog 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 7: Igra „Lavirint“

Autor: Mirko Francuski (mire.etf@gmail.com)
Rukovodilac projekta: Mirko Francuski
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 8: Igra „Potapanje brodova“

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 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 9: Igra „Sudoku“

Autor: Branko Kokanović (branko.kokanovic@gmail.com)
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.