Upute za obavljanje laboratorijskih vježbi

Laboratorijske vježbe na predmetu UTR provode se kroz sustav SPRUT koji omogućuje automatsku evaluaciju programskih rješenja. Sustav za autentikaciju koristi FERWeb, tj. korisničko ime i lozinka jednake su onima na FERWebu. Sustav koristi samopotpisani SSL certifikat tako da možete očekivati upozorenje preglednika da stranica nije sigurna, ali na to se u ovom slučaju nemojte obazirati. Ako se ne možete prijaviti na sustav koristeći svoje FERWeb podatke, što prije se javite na e-mail listu predmeta.

Laboratorijske vježbe mogu se rješavati u programskim jezicima **C, C++, C#, Java i Python (2.x i 3.x)**. Rješenja svih laboratorijskih vježbi trebaju čitati ulazne podatke **sa standardnog ulaza (stdin)**. Točan format ulaznih podataka bit će po retcima definiran za svaku pojedinu vježbu, a svaki redak (uključujući i zadnji) završava znakom za kraj retka. Nadalje, rješenja trebaju izlazne podatke ispisati **na standardni izlaz (stdout) točno onako kako je to definirano u uputama za vježbu**. To među ostalim znači da bilo kakav dodatni ispis (na primjer, kontrolni ispis nekih varijabli) treba biti usmjeren na izlaz za greške (stderr) ili, još bolje, ugašen prije predaje.

Kako biste testirali takva rješenja lokalno, najlakše je ulazne podatke zapisati u tekstualnu datoteku i onda tu tekstualnu datoteku preusmjeriti na ulaz programa. Na primjer, ako je ulaz u zadatak **lab** definicija nekakvog automata, definiciju biste zapisali u neku datoteku **primjer.txt** i pozvali program sa **lab.exe <primjer.txt** (ili nešto slično). Ovakav način pokretanja programa možete podesiti i u svakom IDE-u ili boljem uređivaču teksta.

Rješenje se proizvoljno može sastojati od jedne datoteke s izvornim kodom ili više datoteka (na primjer, nekoliko .h i nekoliko .c datoteka), a sve datoteke s izvornim kodom moraju se nalaziti u istom direktoriju. Ako se rješenje sastoji od jedne datoteke, dovoljno je predati samo tu datoteku. Ako se pak rješenje sastoji od više datoteka, onda je nužno sve potrebne datoteke pohraniti u jednu ZIP arhivu i ta arhiva se predaje na sustav. U ZIP arhivi ne smije biti nikakvih direktorija, tj. datoteke se trebaju nalaziti izravno u korijenu arhive. U arhivu se stavljaju isključivo datoteke s kodom rješenja, bez datoteka s metapodacima Eclipse/VS projekata i slično.

Kada rješenje predate na sustav, sustav će (za prevođene jezike) prevesti rješenje i izvesti ga nad malim skupom ispitnih primjera koje nazivamo *integracijski testovi*. Ovaj skup će tipično sadržavati samo primjer iz upute za vježbu, a ponekad još nekoliko primjera. U svakom slučaju, ako rješenje prolazi integracijske testove, onda možete biti sigurni da

1. ako je rješenje u ZIP arhivi, arhiva je ispravno složena, tj. datoteke su u korijenskom direktoriju
2. rješenje se uspješno prevodi na sustavu
3. rješenje uspješno čita ulazne podatke u integracijskim testovima i daje ispravan ispis za dane integracijske testove

Smisao integracijskih testova su točke 1 i 2 gornje liste, a točka 3 može povećati sigurnost da ste dobro razumjeli format ulaznih podataka i očekivani format izlaza. **Važno** je uočiti da je činjenica da rješenje prolazi integracijske testove vrlo slab indikator da je rješenje stvarno točno. Od vas se očekuje da točnost rješenja provjerite sami na vlastitim primjerima ulaznih podataka i očekivanog izlaza (svakako je dozvoljeno da te primjere međusobno razmjenjujete).

Ako rješenje ne prolazi integracijske testove, od sustava ćete dobiti što je više moguće informacija koje bi vam trebale omogućiti da otkrijete u čemu je problem. Na primjer, ako se rješenje nije uspješno prevelo, dobit ćete izlaz kompilatora. Ako ispis vašeg programa ne odgovara očekivanom ispisu za zadani primjer, vidjet ćete ispis vašeg programa i očekivani ispis. Ako niste u mogućnosti ispraviti svoje rješenje na osnovi

dobivenih informacija, javite se na e-mail listu predmeta. Ipak, **nemojte očekivati da će asistenti ispravljati greške u vašem rješenju.** Ako se ne radi o greški u sustavu ili ispitnim datotekama, vaš je zadatak da svoje rješenje ispravite. Argumenti "rješenje mi radi doma, ali na sustavu ne radi" neće se uzimati u obzir.

Rješenje možete predati na sustav proizvoljan broj puta, sve do isteka roka za predaju koji će biti definiran za svaki zadatak, bez ikakve "kazne" u vidu nekakvih negativnih bodova i slično. Unutar svake minute, rješenje je moguće predati najviše jednom. Vremenski interval za predaju svake vježbe bit će mnogostruko veći od potrebnog vremena za rješavanje vježbe i zato je preporučljivo rješenje prvi puta predati barem nekoliko dana prije krajnjeg roka za predaju. Pod velikim opterećenjem tijekom zadnjih nekoliko sati intervala za predaju, sustav se tipično značajno uspori. Predaja je svejedno moguća, ali nikako nemojte računati da ćete na samom kraju roka dobivati rezultate integracijskih testova i na taj način otklanjati greške u svom rješenju.

Za **C i C++** rješenja, sustav koristi **GCC 4.8.2** (program gcc za C i g++ za C++) uz zastavice **-W -Wall -O2** i dodatno zastavicu **-std=c++0x** za C++ (ova zastavica omogućuje C++11). Imena svih datoteka za rješenja u jezicima C i C++ su proizvoljna, ali očito mora biti točno jedna datoteka s programskim kodom koja definira funkciju **main**. Očekivane ekstenzije su .c za C i .cpp za C++.

Za Javu sustav koristi **javac 1.7.0_75**, a CLASSPATH je podešen na direktorij u kojemu se program nalazi. Ime razreda koji je ulazna točka u rješenje bit će definirano za svaki zadatak. Uočite da organizacija datoteka s programskim kodom u jedan direktorij **onemogućuje korištenje paketa** - sve klase moraju biti u pretpostavljenom paketu (dakle, **ne smiju imati package direktivu**). Iako je to općenito loša praksa za veće projekte, kako se radi o jednostavnim programima, jedan paket bit će sasvim dovoljan. Očekivana ekstenzija za Java programe je .java.

Za **Python 2.x** sustav koristi **Python 2.7.6**, a za **Python 3.x** koristi **Python 3.4.0**. Ulazna točka u rješenje tj. datoteka koja će biti pozvana od strane sustava bit će definirana za svaki zadatak, a očekivana ekstenzija je .py.

Konačno, **C#** sustav prevodi koristeći kompilator **mcs 3.2.8.0** iz **Mono** radnog okvira, a za pokretanje koristi program **mono** iste verzije. Ulazna točka se može nalaziti u bilo kojem razredu, a očekivana ekstenzija je .cs. **Važno** je napomenuti da **nije moguće** predati npr. Visual Studio projekt i očekivati da će takvo rješenje raditi. Datoteke s izvornim kodom treba izvaditi iz projekta i organizirati ih isto kao i za sve ostale jezike. Nadalje, iako Mono podržava praktički sve funkcionalnosti jezika C# i .Net okruženja, ako planirate koristiti nešto neuobičajeno, preporučljivo je da se ukratko upoznate s njegovim ograničenjima ili jednostavno isprobate funkcionalnost na sustavu.

Ako niste sigurni u kojem jeziku rješavati labose, preporuča se korištenje Pythona ili Jave.

Iako se ne očekuje da će sva rješenja biti "optimalne" implementacije nekog algoritma niti da će nužno koristiti optimalan algoritam za zadani zadatak, za izvođenje svakog rješenja bit će definirano maksimalno vrijeme izvođenja. Ovo maksimalno vrijeme izvođenja će biti mnogostruko veće od vremena koje je potrebno referentnom rješenju za najsloženiji ispitni primjer. Drugim riječima, razumne implementacije razumnih algoritama će se izvršiti u daleko kraćem vremenu.

Nakon isteka roka za predaju, sva rješenja bit će evaluirana nad nešto većim brojem ispitnih primjera (tipično 10-30, ovisno o zadatku). U skupu ispitnih primjera nalazit će se primjeri različite složenosti, ali svi primjeri imaju istu težinu tj. "vrijede" jednak broj bodova. Drugim riječima, ako se zadatak evaluira nad n ispitnih primjera, vrijednost svakog točno riješenog primjera je 6/n bodova. Dio ispitnih primjera za pojedine

zadatke bit će objavljen unaprijed u repozitoriju laboratorijskih vježbi i može poslužiti za testiranje programskog rješenja.

Za svaku vježbu postoje **dva roka za predaju**. Na prvom roku moguće je ostvariti maksimalne bodove, dok je na drugom roku, kasnijeg datuma, moguće ostvariti najviše 50% bodova. Detalje o rokovima možete pronaći ovdje.

Konačno, jedna vrlo važna napomena: Nemojte varati, tj. prikazivati tuđe rješenje ili dio tuđeg rješenja kao svoje. Mi ćemo se maksimalno potruditi i automatiziranim i ručnim metodama otkriti i kazniti sve oblike varanja kako bi svi imali jednake uvjete za obavljanje ovih laboratorijskih vježbi. Dozvoljeno je s kolegama raspravljati o idejama i algoritmima koje ćete implementirati, ali bilo kakav oblik dijeljenja koda ili "pseudokoda" je zabranjen. Nemojte ignorirati ovu napomenu!

Kako biste mogli isprobati sustav, trenutno je na sustavu definiran probni zadatak koji je u nastavku i opisan. Ovaj zadatak služi isključivo za upoznavanje sa sustavom i neće se bodovati. Samim time, niste ga obvezni pokušati riješiti, ali moguće je da će vam tada rješavanje laboratorijskih vježbi biti teže. Za čitanje ovog opisa zadatka, te rješavanje samog zadatka i predaju rješenja na sustav sigurno neće biti potrebno više od pola sata.

Program će na ulaz dobiti maksimalno 100 brojeva ne većih od 100, pri čemu se svaki broj nalazi u svom retku. Za svaki broj na ulazu program treba ispisati njegov kvadrat, pri čemu je opet svaki kvadrat u svom retku. Ograničenje na vrijeme izvođenja je 1s (ograničenja za labose će biti znatno veća). Ulazna točka za Java rješenja treba se nalaziti u razredu **proba**, a za Python rješenja u datoteci **proba.py**.

Na primjer, rješenje u jeziku C moglo bi izgledati ovako:

```
#include <stdio.h>
int main(void) {
  int x;
  while (scanf("%d", &x) == 1) {
    printf("%d\n", x*x);
  }
  return 0;
}
```

Moguća rješenja u ostalim jezicima možete vidjeti ovdje.

Pokušajte u rješenje u jeziku po izboru namjerno unijeti neku pogrešku kako biste vidjeli kako će izgledati izlaz sustava u tom slučaju.