Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave
Predmet: **Uvod u teoriju računarstva**

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 (npr. u Google Chromeu treba kliknuti na *Advanced -> Proceed to balrog.zemris.fer.hr (unsafe)*). Ako se ne možete prijaviti na sustav koristeći svoje FERWeb podatke, što prije se javite na email listu predmeta.

Laboratorijske vježbe mogu se rješavati u programskim jezicima C, C++, C#, Java i Python (2.7 i 3.8). 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 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.

Na sustavu SPRUT, ulazni podatci bit će iz datoteke preusmjereni na standardni ulaz (*stdin*) vašeg programa. Program koji očekuje dodatni prazan redak (dodatni “enter”) na kraju ulaznih podataka **neće biti ispravan** ako taj redak nije dio ulaznih podataka, tj. ako ne postoji u testnoj datoteci. Stoga ne preporučujemo testiranje programa “ručnim” unosom (ili copy/pasteom) ulaznih podataka, nego preusmjeravanjem kao u idućem odlomku. (Ako broj redaka ulaznih podataka nije unaprijed zadan, možete ih pročitati npr. while petljom koja provjerava je li redak uspješno unesen, ili provjerava je li pročitan znak za kraj datoteke - EOF).

Kako biste **testirali rješenja lokalno**, potrebno 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 nekog automata, definiciju biste zapisali u neku datoteku `primjer.txt` i pozvali program iz komandne linije sa `lab.exe < primjer.txt` (ili nešto slično). Ovakav način pokretanja programa možete podesiti i u svakom IDEu ili boljem uređivaču teksta. Za više detalja pogledajte odjeljak *Lokalno testiranje* u drugom dijelu dokumenta.

Na sustav SPRUT predaje se datoteka s izvornim kodom ili zip arhiva u slučaju većeg broja datoteka. U arhivu se stavljaju isključivo datoteke s kodom rješenja, bez datoteka s metapodatcima Eclipse/VS projekata i slično. Važno je da pri definiranju putanja u svojim

programima nikada ne koristite znak \ (backslash) kao separator direktorija, nego isključivo znak / (slash). Ovaj separator će raditi i na Windowsima i na Linux računalima kao što je poslužitelj na kojem se izvodi SPRUT. Štoviše, bolja praksa je koristiti knjižnice za formiranje putanja koje su dostupne u svim raspoloživim jezicima, ali to očito nije nužno. **U izvornim kodovima, kao ni u imenima datoteka (uključujući zip arhivu), ne smije biti dijakritičkih znakova.**

Kada rješenje predate na sustav, sustav će 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
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) te na ispitnim primjerima objavljenima u repozitoriju *Laboratorijske vježbe -> Ispitni primjeri*.

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 to ne bude dovoljno, nemojte se ustručavati javiti se na email predmeta, ali tek nakon što pokušate samostalno otkloniti problem. Ipak, ne očekujte da će asistenti proučavati vaše rješenje i tražiti “bugove”. 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 je unaprijed definiran za svaki zadatak. 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 put predati barem nekoliko dana prije krajnjeg roka za predaju. Izbjegavajte predaju na zadnji dan roka, a posebno u zadnjim satima roka jer je tada sustav obično jako opterećen.

Za C i C++ rješenja, sustav koristi GCC 9.3.0 (program `gcc` za C i `g++` za C++) uz zastavice `-std=c99 -O2 -W -Wall` za C, te `-std=c++17 -O2 -W -Wall` za C++. Imena svih datoteka za rješenja u jezicima C i C++ su proizvoljna, ali očito u korijenskom direktoriju 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 11.0.8`, a CLASSPATH je podešen na direktorij u kojemu se program nalazi. Na primjer (za težu inačicu prve vježbe), kada se izvodi generator, CLASSPATH je podešen na korijenski direktorij raspakirane zip arhive, a kada se izvodi analizator, na poddirektorij analizator. Programi ne smiju imati package direktivu.

C# sustav prevodi koristeći kompilator `mcs 6.8.0.105` iz Mono radnog okvira, a za pokretanje koristi program `mono` iste verzije. Ulazna točka može se 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.

Kako izgradnja učinkovitog jezičnog procesora nije središnja tema ovog predmeta, pokušat ćemo ne ocjenjivati brzinu izvođenja, tj. ne kažnjavati sporo izvođenje. Međutim, zbog prirode automatske evaluacije, vremenska ograničenja moraju postojati i ona su obično definirana u uputama za vježbu, ili su postavljena relativno visoko, s visokim stupnjem tolerancije neefikasnih algoritama.

Uzmite u obzir da je postupak evaluacije rješenja relativno složen i ponavlja se od početka za svaki (pod)test. Zbog toga pri predaji budite strpljivi, pogotovo ako znate da je rješenje sporo. Kada dobijete rezultate integracijskih testova, možete biti sigurni da je Vaše rješenje uspješno predano - nije nužno da svi integracijski testovi budu točno riješeni.

Integracijski testovi ne računaju se u konačan rezultat evaluacije, nego će se nakon isteka roka za predaju rješenja ispitati opsežnijim skupom testova. Složenost testova bit će vrlo raznolika, od vrlo jednostavnih do vrlo složenih primjera. Važno je uočiti da integracijski testovi ne služe provjeravanju je li rješenje točno ili nije - oni služe samo kao potvrda da se rješenje može evaluirati na sustavu, tj. da je arhiva dobro oblikovana i slično. Nemojte si dozvoliti da prije provođenja evaluacije sami ne testirate svoje rješenje i da zbog banalnih grešaka gubite bodove.

Konačno, jedna vrlo važna napomena: nemojte varati, tj. prikazivati tuđe rješenje ili dio tuđeg rješenja kao svoje. Nastavnici će se maksimalno potruditi te automatiziranim i ručnim metodama otkriti i kazniti sve oblike varanja. Dozvoljeno je s kolegama raspravljati o idejama i algoritmima koje ćete implementirati, ali bilo kakav oblik dijeljenja koda ili "pseudokoda" je zabranjen. Nemojte zanemariti ovu napomenu!

Lokalno testiranje

Ispitni primjeri nalaze se u repozitoriju *Laboratorijske vježbe -> Ispitni primjeri*. Svaka zip arhiva sadrži direktorije od kojih svaki odgovara jednom testu. Svaki direktorij sadrži dvije tekstualne datoteke od kojih jedna predstavlja ulaz, a druga očekivani izlaz programa.

Prije same predaje vježbe na SPRUT, obavezno ispitajte rad vašeg rješenja nad navedenim ispitnim primjerima. Najpraktičniji način provjere ispravnosti vašeg rješenja je usporedba (liniju po liniju) izlaza programa s očekivanim izlazom koristeći standardni alat naredbenog redka *diff* (Unix).

Npr. ako postoji ispitni primjer *test1* (*t.ul* je ulazna datoteka, *t.iz* je očekivani izlaz), onda izlaz vašeg (npr. python) programa možete ispisati naredbom:

```
$ python3 lab.py < test1/t.ul
```

Dodatno, ako želite, izlaz možete spremiti u datoteku:

```
$ python3 lab.py < test1/t.ul > izlaz1.iz
```

Vaš izlaz (*izlaz1.iz*) uspoređujete s očekivanim izlazom naredbom:

```
$ diff izlaz1.iz test1/t.iz
```

Nadalje, možete izostaviti eksplicitno pisanje izlaza vašeg programa u datoteku te usporediti izlaze jednom naredbom koja usmjerava (“|”) izlaz iz vašeg programa u program *diff* (znak “-” označava standardni ulaz):

```
$ python3 lab.py < test1/t.ul | diff test1/t.iz -
```

Konačno, ako je zadano više ispitnih primjera, usporedbu izlaza možete automatizirati skriptom. Skica takve skripte za 20 ispitnih primjera u direktorijima *test1, test2, ..., test20* prikazana je sljedećim isječkom:

```
#!/bin/bash

for i in {1..20}  # broj ispitnih primjera
do
    # generiraj ime direktorija s vodećom nulom
    dir=$(printf "%0*d\n" 2 $i)
    echo "Test $dir"
    # pokreni program i provjeri izlaz
    res=`python lab.py < test$dir/t.ul | diff test$dir/t.iz -`
    if [ "$res" != "" ]
    then
        # izlazi ne odgovaraju
        echo "FAIL"
        echo $res
    else
        # OK!
        echo "OK"
    fi
done
```

Skriptu spremite u datoteku (npr. *runtests.sh*), omogućite izvršavanje skripte te pokrenite ispitivanje:

```
$ chmod +x runtests.sh
$ ./runtests.sh
Test 01
OK
Test 02
OK
...
```

Probni zadatak

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 1 sekunda (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 pronaći u repozitoriju (*Laboratorijske vježbe -> SPRUT - Proba rješenja).*

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