Pravila polaganja predmeta - NTP

Predmet se može položiti na jedan od dva načina:

- putem pismenog ispita ili
- putem izrade i odbrane projektnog zadatka.

1. Polaganje predmeta putem pismenog testa

Pismeni test se može polagati u dva termina i to:

- u terminu predavanja u nedelji između 15.01.2023 i 19.01.2023.
- u terminu ispita u januarsko-februarskom ispitnom roku.

U oba termina, polagaće se celo gradivo. Da biste predmet položili na ovaj način, potrebno je da se prijavite za test. Prijava za prvi termin testa je putem Google forme (koja će biti blagovremeno objavljena), dok se za drugi termin prijava obavlja putem studentskog veb-servisa. Prijava je obavezna kako bi se alociralo dovoljno prostora za sve studente. Studenti koji se ne prijave neće imati mogućnost izlaska na test.

Pismeni test se realizuje u formi testa na papiru i sastoji iz 3 celine. Prvu celinu čine teorijska pitanja koja se tiču koncepata programskih jezika *Python* i *Rust*. Od studenata neće biti zahtevani opširni odgovori, već će na sva pitanja biti moguće odgovoriti sažeto. Na pojedina pitanja je moguće ostvariti i negativne poene. Da bi se predmet položio neophodno je ostvariti preko 50% poena na teorijskom testu.

Ostale dve celine se sastoje iz praktičnih zadataka koji se tiču *Python* i *Rust* programskih jezika. Zadaci neće zahtevati preterano pisanje koda i u velikoj meri će se poklapati sa zadacima koji su dati u materijalima sa vežbi, uz pojedine izmene.

Napomena: Pismeno polaganje predmeta je moguće u oba termina. Ukoliko niste zadovoljni ocenom sa prvog termina testa, možete je poništiti i izaći na test u drugom terminu.

2. Polaganje predmeta putem izrade projektnog zadatka

Studenti koji nisu zadovoljni zasluženom ocenom sa pismenog testa, a žele da predmet polože u ovoj školskoj godini, mogu se odlučiti za izradu i odbranu projektnog zadatka. Na projektu je moguće ostvariti najviše za dve vrednosti veću ocenu u odnosu na ocenu sa poslednjeg testa koji je student radio. Na

primer, ukoliko je student dobio ocenu 7 na prethodnom testu i želi da poništi ocenu, tada ocena na projektu ne može biti veća od 9. Ukoliko student nije položio test, tada je najveća ocena koju može dobiti na projektu 7.

Procedura prijave za izradu projektnog zadatka podrazumeva nekoliko koraka. Naime, potrebno je da se do <**datuma koji će biti objavljen naknadno**> prijavite za izradu projektnog zadatka, pri čemu treba ispratiti sledeće uputstvo:

- Kreirati privatni repozitorijum na *GitHub*-u.
- Dodati predmetnog asistenta kao saradnika na repozitorijum
    - Dušan Nikolić: *n-dusan* na GitHub-u (smerovi SIIT i SIT),
    - Nikola Nemeš: *NikolaNemes* na *GitHub*-u (smer SIIT),
- Napraviti *README.md* datoteku i na njenom početku **napisati za koju ocenu radite**. Nakon toga, opisati problem koji se rešava, kao i metode koje će biti korišćene za njegovo rešavanje.
- Uneti potrebne informacije u odgovarajući tab tabele (postaviti link do repozitorijuma u odgovarajuću kolonu).
- Od asistenta ćete dobiti povratnu informaciju (najčešće u vidu *issue*-a na repozitorijumu) o tome da li Vam je tema odobrena, kao i savete o eventualnom proširenju, odnosno smanjenju obima projektnog zadatka.

Što se samog izbora teme za projekat tiče, imate dve mogućnosti:

- Izbor **predefinisane teme**, prilikom čega se moraju ispoštovati zahtevi navedeni u **sekciji 2.1**. Neophodno informacije upisati u tab *Predefinisani*.
- **Samostalno definisati temu** (**sekcija 2.2**), pri čemu se mora uzeti u obzir da je obavezna upotreba *Rust* programskog jezika i opciono kombinovati projekat sa programskim jezikom *Python*. Drugih ograničenja nema, osim toga da projekat mora biti zadovoljavajućeg obima. Neophodne informacije upisati u tab *Samostalno definisani*.

Napomena: Prijavom za izradu projektnog zadatka poništava se ocena sa pismenog testa (ukoliko ste prethodno izašli na isti). Termin odbrane projektnog zadatka biće naknadno objavljen. Projekti se rade samostalno. Projekat možete prijaviti nakon drugog termina testa.

2.1. Predefinisane teme za projektni zadatak

Potrebno je odabrati neki od problema iz oblasti računarstva visokih performansi (engl. *High Performance Computing*, skr. *HPC*) po ličnom nahođenju. Primeri nekih problema su:

- problem n-tela (engl. *n-body problem*),
- simulacija dvostrukog klatna (engl. *double pendulum simulation*),
- Kanonov algoritam za množenje matrica,
- dinamika fluida,
- *Navier–Stokes* jednačina,
- parcijalne diferencijalne jednačine,
- problemi linearne algebre,
- manipulacija grafovima,
- itd.

Za izabrani problem, potrebno je uraditi sledeće:

1. (ocena 6) **Rešiti problem upotrebom programskog jezika *Python*:**
    - Implementirati **sekvencijalnu verziju rešenja** upotrebom programskog jezika ***Python***. Rezultat mora biti bar jedna datoteka koja reprezentuje promene stanja modelovanog sistema (po iteracijama, ukoliko je problem rešavan iterativnim metodom).
    - Implementirati paralelizovanu verziju rešenja upotrebom multiprocessing biblioteke programskog jezika Python. Rezultat mora biti bar jedna datoteka koja reprezentuje promene stanja modelovanog sistema (po iteracijama, ukoliko je problem rešavan iterativnim metodom).
2. (ocena 7) **Rešiti problem upotrebom programskog jezika *Rust*:**
    - Implementirati **sekvencijalnu verziju rešenja** upotrebom programskog jezika ***Rust***. Rezultat mora biti bar jedna datoteka koja reprezentuje promene stanja modelovanog sistema (po iteracijama, ukoliko je problem rešavan iterativnim metodom).
    - Implementirati **paralelizovanu verziju rešenja** uz oslonac na **niti** (engl. *threads*). Rezultat mora biti bar jedna datoteka koja reprezentuje promene stanja modelovanog sistema (po iteracijama, ukoliko je problem rešavan iterativnim metodom).
3. (ocena 8) Uraditi **eksperimente jakog i slabog skaliranja,** koji će uporediti dobijeno **ubrzanje paralelizovane *Python* implementacije** rešenja u odnosu na **sekvencijalnu implementaciju upotrebom istog jezika**.

4. (ocena 9) Uraditi **eksperimente jakog i slabog skaliranja** koji će uporediti dobijeno **ubrzanje paralelizovane *Rust* implementacije** rešenja u odnosu na **sekvencijalnu implementaciju upotrebom istog jezika**.
5. (ocena 10) **Vizualizacija rešenja** (po iteracijama, ukoliko je korišćen iterativni model za rešavanje problema) na osnovu prethodno generisanih datoteka, a uz oslonac na ***Rust*** okruženje. Dozvoljena je upotreba grafičkih biblioteka poput *Rerun*.

Pored rednog broja stavke navedena je ocena koja se može zaslužiti izradom iste. Za pomenutu ocenu neophodno je uraditi i stavke sa manjim rednim brojem. Moguće je napraviti i drugačiju kombinaciju stavki od navedene (npr. 1. i 3. za ocenu 7, itd.).

Eksperimente jakog i slabog skaliranja opisati u formi **izveštaja**. Tom prilikom, potrebno je ispoštovati sledeće stavke:

- Navesti tehničke detalje koji se tiču hardverske i softverske arhitekture sistema na kom su rađeni eksperimenti:
  - model procesora, radni takt, organizacija *cache* memorije, broj fizičkih/logičkih jezgara, broj *NUMA node*-ova, itd.
  - tip i količina RAM memorije
  - Operativni sistem
  - Dodatne biblioteke koju su korišćene, kao i njihove verzije
  - ostale informacije koje mogu uticati na rezultate eksperimenata.
- Odrediti procenat sekvencijalnog dela koda koji se po prirodi problema ne može paralelizovati.
- Odrediti procenat paralelnog dela koda koji se može paralelizovati.
- Odrediti teorijske maksimume ubrzanja u skladu sa *Amdalovim*, odnosno *Gustafsonovim* zakonom. Kao veoma koristan izvor informacija može poslužiti sledeći članak.
- Neophodno je generisati bar 4 grafika:
  - jako skaliranje *Python* paralelne implementacije u skladu sa Amdalovim zakonom
  - jako skaliranje *Rust* paralelne implementacije u skladu sa Amdalovim zakonom
  - slabo skaliranje *Python* paralelne implementacije u skladu sa Gustafsonovim zakonom

    - slabo skaliranje *Rust* paralelne implementacije u skladu sa Gustafsonovim zakonom
- Na svakom od prethodno pomenutih grafika, x-osa predstavlja broj procesorskih jezgara, dok y-osa predstavlja ostvareno ubrzanje. Takođe, na svakom grafiku nacrtati liniju teorijskog maksimuma (idealnog skaliranja) i uporediti je sa dobijenim rezultatima.
- Kod eksperimenta slabog skaliranja, objasniti na koji način se manipuliše poslom, tj. kako se modifikacijom parametara postiže konstantan posao po procesorskom jezgru.
- Svaki grafik treba da sledi potporna tabela sa informacijama o srednjem vremenu izvršavanja, standardnoj devijaciji, kao i o eventualnim *outlier*-ima. Kako bi rezultati bili relevantni, za svaku kombinaciju parametara jakog, odnosno slabog skaliranja (broj procesorskih jezgra, veličina problema iskazana odgovarajućom kombinacijom ulaznih argumenata programa, itd.), ekskluzivno izvršiti programsko rešenje 30-ak puta.
- Dodatne informacije koje se tiču jakog i slabog skaliranja možete pronaći u navedenom članku.

Primeri adaptacija predefinisanih tema:

- Fraktalna stabla,
- Paralelizacija K-Means algoritma,
- Računanje determinante matrice primenom Laplasovog razvoja po prvoj vrsti,
- Analiza serijske i parallelne implementacije algoritama baziranih na Monte-Karlo metodi,
- Kanonov algoritam za množenje matrica (primer jednog projekta na ovu temu možete pronaći na linku),
- Problem n-tela (primer jednog projekta na ovu temu možete pronaći na linku),
- Simulacija dvostrukog klatna,
- Dinamika fluida,
- Navier–Stokes jednačina,
- Problem protoka kanala (Channel Flow),
- Random Forest,
- Huffman Coding Algorithm (ocena 6),
- itd.

2.2. Samostalno definisane teme za projektni zadatak

Ukoliko nisu zainteresovani za oblast konkurentnog programiranja, studenti imaju mogućnost da samostalno definišu temu za projektni zadatak. Potrebno je odabrati određeni realni problem, opisati ga, a takođe i dati predlog rešenja istog. Opis problema i predloga rešenja navesti u *README.md* datoteci. Opisati arhitekturu sistema koju planirate koristiti, pri čemu morate uzeti u obzir da se prilikom implementacije komponenti sistema mora iskoristiti *Rust* i opciono programski jezik *Python.*

Primeri samostalno definisanih tema:

- Autogram (Instagram Automator),
- Segmentacija slike upotrebom PSO algoritma,
- Car Sharing Hub,
- Veb aplikacija za pretragu kulturnih dobara,
- Veb aplikacija za pretragu polovnih automobila,
- Veb aplikacija za pretragu manifestacija,
- Simulacija širenja Covid-19 virusa,
- Veb aplikacija za pretragu fudbalera,
- Veb aplikacija za pretragu i preporuku treninga,
- Veb aplikacija za pretragu mobilnih telefona,
- Veb aplikacija za pretragu polovnih automobila,
- Veb aplikacija za pretragu filmova.

Napomena: U slučaju da se odlučite za izradu veb-aplikacija po uzoru na navedene, potrebno je uvrstiti funkcionalnosti koje se tiču autentifikovanih korisnika. Takođe, moraju biti implementirane CRUD operacije za entitete sistema.

Još primera adaptacije predefinisanih kao i samostalno definsanih tema možete pronaći u sledećoj tabeli.

Napomena: Prethodne generacije su umesto *Rust* programskog jezika koristile *Go(lang)* i *Pharo*. Od ove iteracije kursa, *Rust* je obavezan i zamenio je pomenuti *Go(lang)* kao i *Pharo*.

3. Izrada diplomskog rada

Studenti koji su zainteresovani za dodatno zalaganje na ovom predmetu, imaju mogućnost izrade diplomskog rada. Preporuka je da samostalno definišete temu

za projekat, koji ćete odbraniti, kako biste ispunili uslov za izradu diplomskog rada. Prilikom specifikacije projektnog zadatka, potrebno je navesti i eventualna proširenja, koja ćete naknadno implementirati za diplomski. Nakon završetka implementacionog dela, možete započeti sa pisanjem rada. O obimu teme, kao i eventualnim sugestijama na dostavljeni predlog, možete se konsultovati sa profesorom i asistentom.
Studenti koji su se odlučili za neku od predefinisanih tema, imaju mogućnost da brane diplomski rad jedino ako je tema dovoljnog obima ili će biti proširena nakon odbrane projekta. Neophodno je da projekat ima određen doprinos u oblasti, tj. da je nešto urađeno što već nije. O proveri kompleksnosti predefinisane teme, takođe se možete konsultovati sa profesorom ili sa asistentom.
Napomena: Broj mesta za diplomski rad je ograničen. Studenti koji se odluče za izradu diplomskog rada, treba neophodne informacije da upišu u dokument za prijavu koji će biti objavljen od strane profesora. Potrebno je da Vaši **repozitorijumi** budu **javni**, kako bi svi imali uvid u to što radite.

Samostalno definisane teme za projekat koje su prerasle u diplomske radove:

- Analiza i organizacija fajlova i foldera,
- Problem putujućeg trgovca,
- Društvena mreža za muzičare,
- Određivanje najkraćeg puta za dostavu medicinske opreme,
- Transport paketa železničkim saobraćajem,
- Optimizacija organizacije rasporeda nastave (Schedule),
- Survival Simulation,
- Dizajn i implementacija softvera za analizu Smalltalk programskog koda,
- ClinicHub – portal za klinike baziran na mikroservisnoj arhitekturi,
- GoCinema - Portal za bioskope baziran na mikroservisnoj arhitekturi,
- Protočna obrada fotografija u programskom jeziku Go(lang),
-

Napomena: Dokument će se po potrebi ažurirati.