Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Programiranje u realnom vremenu
Nastavnik: Prof. dr Dragan Milićev
Asistent: Bojan Furlan
Ispitni rok: Jun 2011.
Datum: 19.06.2011.

Kandidat: _____

Broj Indeksa: _____ *E-mail:*_____

Ispit traje 3 sata. Dozvoljeno je korišćenje literature.

Zadatak 1	_____/10	*Zadatak 4*	_____/20
Zadatak 2	_____/10	*Zadatak 5*	_____/10
Zadatak 3	_____/20	*Projekat*	_____/40
Ukupno na ispitu:	**_____/70**	***Ukupno na projektu:***	**_____/40**

Ukupno: _____/110

Ocena: _____ (_____)

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

1. (10 poena) Tolerancija otkaza

Implementirati klasu `List` koja predstavlja jednostruko ulančanu listu sa operacijama stavljanja elementa tipa `Data` na kraj liste i uzimanja elementa sa početka liste. Implementacija treba da ima povećanu otpornost na otkaze tako što može da detektuje određene greške u svojoj strukturi na sledeći način. Svaki ulančani zapis u listi, pored ostalog, sadrži i podatak koji pokazuje kojoj listi pripada taj zapis. Ukoliko se detektuje korupcija tog podatka, operacija koja je detektuje treba da baci izuzetak pozivaocu, bez izmene u strukturi.

Rešenje:

2. (10 poena) Konkurentno programiranje

Na jeziku Ada implementirati ograničeni bafer kao proces sa ulazima (*entry*) za stavljanje i uzimanje elementa iz bafera. Ilustrovati upotrebu ograničenog bafera prikazom koda procesa proizvođača.

Rešenje:

3. (20 poena) Realno vreme

Neki sistem treba da obezbedi pravovremenu reakciju (odziv) na neki sporadični događaj. Vreme reakcije sistema R na taj događaj ne sme da bude ni preuranjeno, ni zakasnelo, već treba da bude u zadatom opsegu: $t_{min} \leq R \leq t_{max}$. Događaj se signalizira postavljanjem određenog registra koji procesor može pročitati. Softver ovog sistema implementiran je kao jedan periodični proces koji je stalno aktivan i koji radi na sledeći način. Ako u nekoj aktivaciji prvi put pronađe da se događaj dogodio, ne radi ništa u toj i u još nekoliko narednih aktivacija (ukupno njih n), kako bi obezbedio protok vremena od bar t_{min}, a onda u n+1-oj aktivaciji reaguje.

Na raspolaganju su sledeće funkcije interfejsa prema uređajima:

`hasEventHappened():Boolean`	Vraća True ako je detektovan događaj.
`react()`	Obavlja reakciju.

(a)(10) Na jeziku Ada implementirati ovaj proces tako da ne pati od kumulativnog plivanja (engl. *cumulative drift*) i ima zaštitu od prekoračenja vremenskog roka, uzimajući vremenske parametre kao simboličku konstantu.

(b)(10) Napisati i objasniti nejednakosti koje predstavljaju uslove za periodu (T), vremenski rok (D) i broj aktivacija n u funkciji t_{min} i t_{max}, a potom odrediti ove parametre (T, D i n), ako je t_{min} = 4,3s, a t_{max} = 4,5s.

Rešenje:

4. (20 poena) Raspoređivanje i rasporedivost

(a)(10) Dat je sledeći skup nezavisnih periodičnih procesa sa $D = T$ koji se raspoređuju po EDF (*Earliest Deadline First)*:

Proces	T	C
a	3	1
b	4	1
c	6	2
d	12	1

Nacrtati vremenski dijagram raspoređivanja ovog skupa procesa počev od kritičnog trenutka i ispitati njihovu rasporedivost. Ukoliko u datom trenutku više procesa ima isti vremenski rok, prednost ima onaj proces koji se do tada izvršavao.

(b)(10) Data su tri periodična, nezavisna procesa sa sledećim karakteristikama:

Proces	T	D	C
a	4	3	1
b	6	2	2
c	18	10	5

Procesima dodeliti prioritete po DMPO i na vremenskom dijagramu prikazati kako se ovi procesi raspoređuju po FPS šemi počev od kritičnog trenutka.

Rešenje:

5. (10 poena) ROOM

Na jeziku ROOM realizovati klasu aktera koji obavlja periodično očitavanje A/D konvertora tehnikom pomeranja periode (*period displacement*) i svaku očitanu vrednost šalje na svoj izlazni port. Sa drugog svog porta ovakav akter može da prima signale za uključivanje, isključivanje i promenu periode očitavanja na novu zadatu vrednost. Na raspolaganju su funkcije za pokretanje konverzije `startAD()` i očitavanje konvertovane vrednosti `readAD()` koje se mogu koristiti na nivou detalja.

Rešenje: