Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet:	Operativni sistemi 1 (SI2OS1, IR2OS1)
Nastavnik:	prof. dr Dragan Milićev
Odsek:	Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum:	Treći, septembar 2016.
Datum:	4.9.2016.

Treći kolokvijum iz Operativnih sistema 1

Kandidat: ____

Broj indeksa: ________________ *E-mail:*______________________________________

Kolokvijum traje 1,5 sat. Dozvoljeno je korišćenje literature.

Zadatak 1	_______________/10	*Zadatak 3*	_______________/10
Zadatak 2	_______________/10		

Ukupno: __________/30 = __________% = _________/10

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) Ulaz/izlaz

Dat je proceduralni interfejs prema nekom blokovski orijentisanom ulaznom uređaju sa direktnim pristupom:

```
extern const int BlockSize;
extern int BlockIOHandle;
long getSize(BlockIOHandle handle);
int readBlock(BlockIOHandle handle, long blockNo, char* addr);
```

Uređaj se identifikuje „ručkom“ tipa `BlockIOHandle`, a blok je veličine `BlockSize` znakova. Operacija `getSize` vraća ukupnu veličinu sadržaja (podataka) na uređaju (u znakovima), a operacija `readBlock` učitava blok sa zadatim brojem u bafer na zadatoj adresi u memoriji i vraća 0 u slučaju uspeha. Obe operacije vraćaju negativnu vrednost u slučaju greške, uključujući i pokušaj čitanja bloka preko granice veličine sadržaja.

Korišćenjem ovog interfejsa implementirati sledeći objektno orijentisani interfejs prema ovom uređaju, koji od njega čini apstrakciju ulaznog toka, odnosno znakovno orijentisanog ulaznog uređaja sa direktnim pristupom:

```
class IOStream {
public:
  IOStream (BlockIOHandle d);
  int seek (long offset);
  int getChar (char& c);
};
```

Operacija `seek` postavlja poziciju „kurzora“ za čitanje na zadatu poziciju (pomeraj počev od znaka na poziciji 0), a operacije `getChar` čita sledeći znak sa tekuće pozicije kurzora u izlazni argument `c` i pomera kurzor za jedno mesto unapred. U slučaju bilo kakve greške, uključujući i pomeranje kurzora preko veličine sadržaja ili čitanje znaka kada je kurzor stigao do kraja sadržaja, operacije treba da vrate negativnu vrednost, a nulu u slučaju uspeha.

Rešenje:

2. (10 poena) Interfejs fajl sistema

Dole je dat izvod iz dokumentacije za API prema znakovnim tokovima (*characted streams*) vezanim za fajlove u GNU sistemima. Sve date deklaracije su u `<stdio.h>`. Korišćenjem samo dole datih funkcija, realizovati sledeću funkciju koja prepisuje ceo sadržaj datog ulaznog fajla u dati izlazni fajl, ali u obrnutom redosledu (poretku) znakova. U slučaju bilo kakve greške, funkcija treba da vrati negativnu vrednost, u suprotnom treba da vrati 0.

```
    int fcopyreverse (const char *filenamefrom, const char *filenameto);
```

`FILE* fopen (const char *filename, const char *opentype);`
Opening a file with the `fopen` function creates a new stream and establishes a connection between the stream and a file. This may involve creating a new file. The `fopen` function opens a stream for I/O to the file `filename`, and returns a pointer to the stream. If the open fails, `fopen` returns a null pointer. The `opentype` argument is a string that controls how the file is opened and specifies attributes of the resulting stream. It must begin with one of the following sequences of characters:
'r' Open an existing file for reading only.
'w' Open the file for writing only. If the file already exists, it is truncated to zero length. Otherwise a new file is created.
'a' Open a file for append access; that is, writing at the end of file only. If the file already exists, its initial contents are unchanged and output to the stream is appended to the end of the file. Otherwise, a new, empty file is created.
'r+' Open an existing file for both reading and writing. The initial contents of the file are unchanged and the initial file position is at the beginning of the file.
'w+' Open a file for both reading and writing. If the file already exists, it is truncated to zero length. Otherwise, a new file is created.
'a+' Open or create file for both reading and appending. If the file exists, its initial contents are unchanged. Otherwise, a new file is created. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

`int fclose (FILE *stream);`
This function causes `stream` to be closed and the connection to the corresponding file to be broken. Any buffered output is written and any buffered input is discarded. The `fclose` function returns a value of 0 if the file was closed successfully, and `EOF` if an error was detected. It is important to check for errors when you call `fclose` to close an output stream, because real, everyday errors can be detected at this time. For example, when `fclose` writes the remaining buffered output, it might get an error because the disk is full. Even if you know the buffer is empty, errors can still occur when closing a file if you are using NFS.

`int fseek (FILE *stream, long int offset, int whence);`
The `fseek` function is used to change the file position of the stream `stream`. The value of `whence` must be one of the constants `SEEK_SET`, `SEEK_CUR`, or `SEEK_END`, to indicate whether the `offset` is relative to the beginning of the file, the current file position, or the end of the file, respectively. This function returns a value of zero if the operation was successful, and a nonzero value to indicate failure. A successful call also clears the end-of-file indicator of stream and discards any characters that were "pushed back" by the use of `ungetc`. `fseek` either flushes any buffered output before setting the file position or else remembers it so it will be written later in its proper place in the file.

`int fgetc (FILE *stream);`
This function reads the next character as an `unsigned char` from the `stream` stream and returns its value, converted to an `int`. If an end-of-file condition or read error occurs, `EOF` is returned instead.

`int fputc (int c, FILE *stream);`
The `fputc` function converts the character `c` to type `unsigned char`, and writes it to the `stream` stream. `EOF` is returned if a write error occurs; otherwise the character `c` is returned.

Rešenje:

3. (10 poena) Implementacija fajl sistema

Neki fajl sistem primenjuje indeksiranu alokaciju fajlova, sa indeksima u dva nivoa. U FCB fajla nalazi se polje `index` tipa `unsigned long`, koje predstavlja broj bloka na disku u kome se nalazi indeks prvog nivoa. Indeksi oba nivoa su iste veličine `IndexSize` ulaza tipa `unsigned long`. Nepopunjeni ulazi u indeksima imaju vrednost 0. Operacija `getPBlock` po potrebi u keš učitava blok sa diska sa zadatim brojem i vraća pokazivač na mesto u kešu u koje je taj blok učitan; u slučaju greške vraća 0.

```
typedef unsigned long ulong;
extern const ulong IndexSize;
void* getPBlock(ulong pBlockNo);
```

Realizovati funkciju `getFileBlock()` datu dole, koja se koristi u implementaciji fajl sistema i koja treba da vrati adresu na učitan logički blok fajla sa zadatim (logičkim) brojem; u slučaju greške treba da vrati 0.

```
void* getFileBlock (FCB* fcb, ulong lBlockNo);
```

Rešenje: