

Ponovljeni završni ispit iz Programiranja i programskog inženjerstva
7.2.2006.

Napomene uz sve zadatke:

- nije dopušteno korištenje globalnih varijabli
- nije dopušteno korištenje goto naredbi
- u svakom zadatku navesti samo neophodne naredbe za uključivanje prototipova funkcija iz standardne biblioteke potprograma

1. Napisati funkciju `jestSigurna` koja vraća **logičku** vrijednost *istina* ako je zadana lozinka sigurna, a **logičku** vrijednost *laž* inače. Sigurna lozinka je znakovni niz koji zadovoljava sva tri navedena uvjeta:

- u sebi ima **barem jedno** slovo abecede (može biti malo, a može biti i veliko)
- u sebi ima **barem jednu** znamenku
- u sebi ima **barem jedan** specijalni znak. Specijalni znakovi su oni koji se mogu ispisati, a nisu ni znamenke ni slova. Koristiti funkciju `isprint` sa službenog podsjetnika

Npr. za lozinku "pero" funkcija vraća *laž*, za "pero18" vraća *laž*, za "pero?18" vraća *istina*. **(700 bodova)**

```
#include <string.h>
#include <ctype.h>

int jestSigurna(char *lozinka) {
    int i;
    int brojSlova = 0, brojZnam = 0, brojPrint = 0;
    for (i = 0; i < strlen(lozinka); i++) {
        if (isalpha(lozinka[i]))
            brojSlova++;
        else if (isdigit(lozinka[i]))
            brojZnam++;
        else if (isprint(lozinka[i]))
            brojPrint++;
    }
    if (brojSlova && brojZnam && brojPrint)
        return 1;
    else
        return 0;
}
```

2. Postojeća neformatirana (binarna) datoteka "stara.dat" sadrži zapise o radnicima:

imeRadnika	20+1 znak
prezRadnika	30+1 znak
placa	realan broj standardne preciznosti

Napisati program koji će sve zapise iz datoteke "stara.dat" prepisati u **novu** formatiranu (tekstualnu) datoteku "nova.dat", tako da se za svaki zapis iz datoteke "stara.dat", po jedan redak upiše u datoteku "nova.dat", npr:

```
Pero Horvat 4200.45
Ana Novak 5020.50
Ivo Ban 3300.82
```

Ukoliko se datoteka "stara.dat" ne može otvoriti ili se datoteka "nova.dat" ne može stvoriti, na zaslon ispisati odgovarajuću poruku i završiti program. **(700 bodova)**

```

#include <stdio.h>
#include <stdlib.h>

int main () {
    FILE *stara, *nova;
    struct {
        char imeRad[20+1];
        char prezRad[30+1];
        float placa;
    } zapRadnik;

    if ((stara = fopen("stara.dat", "rb")) == NULL) {
        printf("Ne mogu otvoriti stara.dat\n");
        exit(1);
    }
    if ((nova = fopen("nova.dat", "w")) == NULL) {
        printf("Ne mogu otvoriti nova.dat\n");
        fclose(stara);
        exit(1);
    }

    while (fread(&zapRadnik, sizeof(zapRadnik), 1, stara) == 1) {
        fprintf(nova, "%s %s %.2f\n",
            zapRadnik.imeRad, zapRadnik.prezRad, zapRadnik.placa);
    }

    fclose(stara);
    fclose(nova);
    return 0;
}

```

3. Napisati funkciju **prijepis** koja elemente zadanog cjelobrojnog jednodimenzionalnog polja, redom s lijeva prema desno, prepisuje u zadani niz znakova. Za zadani niz znakova je u glavnom programu rezervirano dovoljno prostora u memoriji. Za prepisivanje cijelih brojeva u zadani niz znakova **obavezno** koristiti funkciju `sprintf` čiji je prototip definiran u biblioteci "stdio.h" te glasi:

```

int sprintf (char *odredisniZnakovniNiz,
             const char *format, arg1, arg2, ..., argn);

```

Osim toga, funkcija **prijepis** mora vratiti vrijednosti najmanjeg i najvećeg elementa polja. Može se pretpostaviti da zadano cjelobrojno polje sadrži barem jedan element. Npr. za cjelobrojno polje: 1, 53, -5, 7; funkcija vraća cjelobrojne vrijednosti -5 za najmanji i 53 za najveći element, a u zadani niz upisuje "153-57". **(800 bodova)**

Varijanta 1:

```

#include <stdio.h>
void pretvori (int *polje, int n, char *niz, int *najm, int *najv) {
    int i, ispisanoZnakova;
    *najv = polje[0];
    *najm = polje[0];
    for (i = 0; i < n; i++) {
        if (polje[i] > *najv) *najv = polje[i];
        if (polje[i] < *najm) *najm = polje[i];
        ispisanoZnakova = sprintf(niz, "%d", polje[i]);
        niz += ispisanoZnakova;
    }
}

```

Varijanta 2:

```
#include <stdio.h>
#include <string.h>
void pretvori (int *polje, int n, char *niz, int *najm, int *najv) {
    int i;
    char pomocniNiz[12];
    *najv = polje[0];
    *najm = polje[0];
    niz[0] = '\\0';
    for (i = 0; i < n; i++) {
        if (polje[i] > *najv) *najv = polje[i];
        if (polje[i] < *najm) *najm = polje[i];
        sprintf(pomocniNiz, "%d", polje[i]);
        strcat(niz, pomocniNiz);
    }
}
```

4. Napisati funkciju **transp** koja transponira zadanu cjelobrojnu matricu (funkcija **mijenja** zadanu matricu, ne stvara novu). Transponiranje je matematička operacija koja matrici zamjenjuje elemente pojedinog retka i stupca: prvi stupac postaje prvi redak i obrnuto. Npr.

matricu $\begin{bmatrix} 1 & -1 & 2 \\ 2 & 3 & -2 \end{bmatrix}$ transponira u $\begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 2 & -2 \end{bmatrix}$, matricu $\begin{bmatrix} 1 & -1 \\ 2 & 3 \\ 4 & 2 \\ 3 & -2 \end{bmatrix}$ transponira u $\begin{bmatrix} 1 & 2 & 4 & 3 \\ -1 & 3 & 2 & -2 \end{bmatrix}$

Napomena: pretpostaviti da je u glavnome programu matrica definirana s dimenzijama koje su dovoljne za pohranu i ulazne i izmijenjene (transponirane) matrice. (800 bodova)

3. STARO

Napisati funkciju **pretvori** koja za ulazno jednodimenzionalno polje cijelih brojeva stvara znakovni niz u kojem su elementi polja upisani redom s lijeva prema desno. Za stvaranje znakovnog niza **obavezno** koristiti funkciju `sprintf` čiji je prototip definiran u biblioteci "stdio.h" te glasi:

```
int sprintf (char *odredisniZnakovniNiz,
             const char *format, arg1, arg2, ..., argn);
```

Funkcija **pretvori** mora vratiti vrijednosti najmanjeg i najvećeg elementa polja, te novostvoreni znakovni niz. Npr. za cjelobrojno polje: 1, 53, -5, 7; funkcija vraća znakovni niz "153-57", te cjelobrojne vrijednosti -5 za najmanji i 53 za najveći element. Može se pretpostaviti da ulazno polje sadrži barem jedan element. (800 bodova)

```
void transp (int *mat, int *m, int *n, int maxstup) {
    int i, j, maxdim, pom;
    maxdim = *m > *n ? *m : *n;
    for (i = 0; i < maxdim-1; i++)
        for (j = i+1; j < maxdim; j++) {
            pom = mat[i*maxstup + j];
            mat[i*maxstup + j] = mat[j*maxstup + i];
            mat[j*maxstup + i] = pom;
        }
    /* ova zamjena dimenzija se ne zahtijeva, tako da je rjesenje
       u kojem se koriste formalni argumenti int m, int n,
       takodjer ispravno */
    pom = *m;
    *m = *n;
    *n = pom;
}
```