Задача 1

Да се напише програма во C која работи со процеси и нитки. Главната програма (родител процесот) треба да креира дете процес, на кого ќе му прати низа од 100 цели броеви. Дете процесот најпрво треба да ја пополни низата од 100 цели броеви со нули. Потоа, дете процесот треба да креира N нитки (N се внесува од тастатура во родител процесот), притоа на секоја нитка дете процесот и испраќа (како аргумент) случаен позитивен цел број „K“ (најмногу 500). Секоја нитка прави „K“ промени во низата и потоа завршува со работа. Секоја една промена во низата значи случајно одбирање на еден елемент од низата и менување на неговата вредност. Првата половина од нитките ја менуваат вредноста на елементот со зголемување на неговата вредност за 1, додека пак втората половина на нитките ја намалуваат вредноста на елементот за 1. Откако ќе завршат со работа сите нитки, главната нитка (дете процесот) печати на екран колку елементи од низата ја имаат променето својата вредност (т.е. не се повеќе нула). Родител процесот завршува откако дете процесот ќе заврши. Генерирањето на случајни броеви се прави со помош на функцијата rand(). **БОНУС:** Проверката колку елементи од низата ја имаат променето својата вредност да ја прави родител процесот.

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <time.h>

int niza[100];//dostapna za site nitki

void *zgolemi(void *x){
	int k = *((int *)x);
	int i,j;
	for(i = 0; i < K; i++){
		j = rand()%100;//slucajno odbiranje eden element od nizata!
		niza[j]++;
	}
	pthread_exit(NULL);
}

void *namali(void *x){
	int K = *((int *)x);
	int i, j;
	for(i = 0; i < K; i++){
		j = rand()%100;
		niza[j]--;
	}
	pthread_exit(NULL);
}

int main(){
	pid_t dete;
	int N, i;
```

```
	int K[100];
	pthread_t nitki[100];

	printf("Vnesi go N:\n");
	scanf("%d", &N);
	dete = fork();
	if(dete == 0){//vo dete procesot sme
		for(i = 0; i < 100; i++){
			niza[i]=0;//popolnuvanje na nizata so nuli
		}
	for(i = 0; i < N; i++){//kreirame nitki
		K[i] = rand()%500;
		if(i < N/2){//prvata polovina
			pthread_create(&nitki[i], NULL, zgolemi, (void*)&K[i]);
		}
		else{//vtora polovina
			pthread_create(&nitki[i], NULL, namali, (void *)&K[i]);
		}
	}
	//cekame nitkite da zavrsat
	for(i = 0; i < N; i++){
		pthread_join(nitki[i], NULL);
	}

	//treba da prebroime kolku elementi ja menale vrednosta!
	int kolku = 0; //brojac
	for(i = 0; i < 100; i++){
		if(niza[i]!=0){
			kolku++;
		}
	}
	printf("Promeneti se %d elementi\n", kolku);
	}
	else{//vo roditel sme
		wait(NULL); //cekaj da zavrsi
		printf("Zavrsi deteto!");
	}
	return 0;
}
```

Задача 2

Да се напише програма во C која работи со процеси и нитки. Главната програма (родител процесот) добива низа од наредби кои треба да ги изврши како аргументи од командна линија. Родител процесот треба, најпрво, да креира онолку деца процеси колку што има наредби наведено од командна линија (наредбите се без аргументи). Пота, треба да креира онолку нитки, колку што има наредби, така што, секоја нитка ќе чека и ќе брои колку секунди му било потребно на соодветниот процес да заврши. Тоа значи дека, првата нитка ќе биде задолжена за првата наредба т.е. за првиот процес, втората за вториот и т.н. Секоја нитка брои колку време се извршувал нејзиниот процес (наредба) и кога ќе заврши кажува колку вкупно секунди му требало да заврши, а потоа и

самата нитка завршува. Откако ќе завршат сите процеси/нитки, тогаш главниот процес/нитка печати на екран колку време и требало на секоја наредба да се изврши.
БОНУС: главниот процес/нитка печати на екран колку време и требало на секоја наредба да се изврши во растечки редослед според времето на завршување, најпрво да се отпечати која наредба и требало најмалку време да заврши, па потоа следната наредба и т.н.

```
#include <stdio.h>#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <time.h>

pid_t deca[100];//globalna, nitkite da imaat pristap
int sekundi [100];

void *broi(void *x){
	int i = *((int *)x);
	sekundi[i]=0;
	while(1){
		if(deca[i]==waitpid(deca[i], NULL, WNOHANG)){
			break;
		}
		sleep(1);
		sekundi[i]++;
	}
	pthread_exit(NULL);
}


int main(int argc, char *argv[]){
	//kolku naredbi, tolku deca treba da se kreiraat, potrebno e da go najdeme brojot na naredbi
	int n = argc - 1;
	int i,j;
	int K[10];
	pthread_t nitki[10];
	int pom;
	char poms[100];

	for(i = 0; i < n; i++){
		deca[i] = fork();
		//treba da sprecime kreiranje deca
		if(deca[i]==0){break;}
	}
	if(i != n){//vo dete proces sme
		execlp(argv[i+1], argv[i+1], NULL);
		printf("Neuspesen povik na %s\n", argv[i+1]);
	}
```

```
	else{//kreirame nitki
		for(i = 0; i < n; i++){
			K[i] = i;
			pthread_create(&nitki[i], NULL, broi, (void *)&K[i]);
		}
	//cekame da zavrsat
		for(i = 0; i < n; i++){
			pthread_join(nitki[i], NULL);
		}
	//bonus
		for(i = 0; i < n; i++){
			for(j = i+1; j < n; j++){
				if(sekundi[i]>sekundi[j]{
					pom = sekundi[i];
					sekundi[i]=sekundi[j];
					sekundi[j]=pom;
					strcpy(poms, argv[i]);
					strcpy(argv[i], argv[j]);
					strcpy(argv[j], poms);
				}
			}
		}
	//pecatenje kolku vreme i trebalo na sekoja nitka da zavrsi
		for(i = 0; i < n; i++){
			printf("Naredba %s, %d sekundi\n", argv[i+1], sekundi[i]);
		}
	}
	return 0;
}
```

Да се напише С програма за кодирање на дадена слика од страна на „N“ нитки. Секоја нитка, како аргументи добива: <koj del od slikata go kodira> < kolku pikseli ima toj del>. Сликата е запишана во текстуална датотека, така што секој знак е пиксел кој има код на боја од 0 до 255. Секоја нитка паралелно врши кодирање на сликата, притоа секоја си знае кој дел е нејзин и колкаво парче (ги добива горните леви координати на подматрицата која треба да ја кодира и димензијата на подматрицата). Кодирањето се прави така што се зема пиксел по пиксел и се проверува дали неговите соседни пиксели (околу него во подматрицата) имаат иста вредност или се разликуваат најмногу за 1. Доколку е тој услов исполнет, тогаш на екран процесот ги печати координатите на пикселот и бојата со која се кодира. Доколку не е исполнет условот, на екран не се печати ништо. Сликата е со димензии 1096x1080. Главната нитка ја пополнува матрицата од датотеката и ги започнува останатите нитки. Бројот на нитки со кои се кодира се внесува како аргумент од командна линија, додека пак, податоците што се праќаат на секоја нитка (како аргументи) се внесуваат од тастатура од главната нитка.

```
#include <stdlib.h>
```

```
#include <pthread.h>
#include <ctype.h>
#include <time.h>
#include <math.h>

typedef struct{
    int x, y;
    int size;
}data;

int matrica[1096][1080];

int check(int a, int b){
    return abs(a-b)>0;
}

void *funk (void *x){
    data d = *((data *)x);//dereferenciranje
    int i, j, proverka;
    for (i = d.x; i < d.x + d.size; i++){
        for(j = d.y; j < d.y + d.size; j++){
            proverka = 1;
            if(i == d.x){
                //prva redica
                if(check(matrica[i][j], matrica[i][j+1]))
                {   proverka = 0;
                }
                if(j != d.y + d.size-1 && j!=0)
            {
                                if(check(matrica[i][j],   matrica[i+1][j+1])||
check(matrica[i][j], matrica[i][j+1])){
                    proverka = 0;
                }
                else if(j != 0){
                                 if(check(matrica[i][j],  matrica[i+1][j])  ||
check(matrica[i][j], matrica[i][j-1])){
                        proverka = 0;
                    }
                }

            } else{
                if (check(matrica[i][j], matrica[i+1][j+1])){
                    proverka = 0;
                }
            }
            if(proverka == 1){
                printf("mat[%d][%d] = %d", i, j, matrica[i][j]);

            }
        }
```

```
    }
    pthread_exit(NULL);
}

int main(int argc. char *argv[]){
    pthread_t nitki[100];
    int N;
    data d[100];
    int i,j;
    if(argc < 3){
        printf("nedovolno argumenti");
        return 0;
    }
    FILE *fp = fopen(argv[1], 'r');
    if(fp == NULL){
        printf("Ne moze da se otvori!");
        return 0;
    }

    for(i = 0; i < 1096; i++){
        for(j = 0; j < 1080; j++){
            matrica[i][j] = fgetc(fp);
        }
    }
    fclose(fp);
    N = atoi[argv[2]];
    for(i = 0; i < N; i++){
        printf("Vnesi gi koordinatite na podmatricata i dimenzijata: \n");
        scanf("%d %d %d", &d[i].x, &d[i].y, &d[i].size);
        pthread_create(&nitki[i], NULL, funk, (void *)(d+i));
    }

    for(i = 0; i< N; i++){
        pthread_join(nitki[i], NULL);
    }
    printf("Zavrsija nitkite so kodiranje\n");

    return 0;
}
```

Задача3

Да се напише програма во C која работи со процеси. Програмата (главниот ппроцес) треба да дозволи да се внесе целобројна вредност од тастатура K (најмногу 100). Потоа, главниот процес треба да дозволи да се внесат K наредби од тастатура, притоа секоја наредба што се внесува има име на наредбата и три аргументи. Главниот процес, за секоја наредба внесена од тастатура треба да креира дете процес, на кое ја испраќа наредбата (заедно со трите аргументи), притоа, дете процесот треба да ја изврши таа наредба. Секое од деца процесите, пред да ја изврши наредбата, одбира случаен број од 1 до 20

(со наредбата rand()%20+1) и чека толку секунди пред да почне да ја извршува наредбата. Дете процесот печати колку секунди ќе чека пред да почне со чекање. Главниот процес, ги чека деца процесите да завршат со извршување, потоа печати на екран дека и тој завршува.
БОНУС: Главниот процес да измери колку секунди им требало на сите деца да завршат со извршување.

задача3

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<time.h>
#include<math.h>
#include<sys/wait.h>

int main(){
	int K, i;
	//deklaracija na naredba i argumenti
	char n[30], a1[30], a2[30], a3[30];
	pid_t dete[30];

	printf("Vnesi go K (K<=100)\n");
	scanf("%d", &K);

	for(i = 0; i < K; i++){
		printf("Vnesi naredba, pa argumenti: \n");
		scanf("%s %s %s %s", &n, &a1, &a2, &a3);
		dete[i] = fork();
		if(dete[i] == 0){ break; }
	}
	if(i < K){//dete proces, samo K e roditel
		int cekaj;
		cekaj = rand()%20+1;
		printf("Dete %d ceka %d sekundi ", i, cekaj);
		sleep(cekaj);
		execlp(n, n, a1, a2, a3, NULL);
		printf("Neuspesen povik na dete %d!", i);
	}
	else{//vo roditelot sme
		//bonus
		int sekundi = 0;
		int procesi[K];
		for(i = 0; i < K; i++){
			procesi[i]=-1;}
		while (1){
			for(i = 0; i < K; i++){
				procesi[i] = waitpid(deca[i], NULL, WNOHANG);
			}
			for(i = 0; i < K; i++){
				if(procesi[i]==-1){break;}
			}
```

```
            if(i==K){break;}
            sleep(1);
            sekundi++;
            }
            print("Zavrsuva i roditelot!");
            for(i = 0; i < K; i++){
                  wait(NULL);
            }
            printf("Zavrsija decata\n");
      }
      return 0;
}
```

Задача4

Да се напише програма во C која работи со процеси и нитки. Главната програма (родител процесот) пополнува низа од 1000 броеви на случаен начин со помош на rand() функцијата. Потоа, креира два деца процеси, така што, двата деца процеси вршат пребарување на бројот 0 во низата од 1000 броеви. Првото дете процес, пребарувањето го прави со помош на 10 деца процеси, додека пак второто дете, пребарувањето го прави со 10 нитки. Секоја нитка/дете процес добива дел од низата што треба да го пребара бројот 0 и печати на екран колку пати е пронајден бројот кај соодветната нитка/дете процес. Родител процесот чека да завршат двете деца процеси и на екран печати кое од двете завршило прво. Доколку прво заврши дете процесот кое пребарувањето го прави со помош на деца процеси, тогаш на екран се печати „pobrzo zavrshi prebaruvanjeto so deca procesi“, инаку печати „pobrzo zavrsi prebaruvanjetopobrzo zavrshi prebaruvanjeto so deca procesi“, инаку печати „pobrzo zavrshi prebaruvanjeto so deca procesi“, инаку печати „pobrzo zavrsi prebaruvanjetopobrzo zavrsi prebaruvanjeto so nitki“.

БОНУС: Да се имплементира кодот така што двете деца процеси печатат само по еднаш колку вкупно е пронајдена 0. Кај првото дете процес, комуникацијата со децата процеси кои пребаруваат да се прави со помош на датотека со име „pobrzo zavrshi prebaruvanjeto so deca procesi“, инаку печати „pobrzo zavrsi prebaruvanjetocommunicate.txt“.

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <stdlib.h>


#DEFINE N 1000
int niza[N];
int nulivonitki[10];

void *prebaraj(void *x){
      int i = *((int *)x);
      int j, found = 0;
      for(j = i*100; j < i*100+100; j++){
```

```
            if(niza[j]==0){found++;}
      }
      nulivonitki[i] = found;
      pthread_exit(NULL);
}

int main(){
      pid_t deteProcesi, deteNitki;
      pid_t deca[10]//za deteProcesi
      pthread_t nitki[10];
      int i,j;
      int vkupno1=0, vkupno2=0, nuli1=0;
      int thread_data[10];

      for(i = 0; i < N; i++){
            niza[i]=rand()%10;
      }
      deteProcesi = fork();
      if(deteProcesi == 0){
            for(int i = 0; i < 10; i++){
                  deca[i]=fork();
                  if(deca[i]==0){break;}//da ne kreiraat decata deca!
            }
            if(i < 10){//vo nekoe od deca procesite sme!
                  for(j = i*100; j < i*100+100; j++){
                        if(niza[j] == 0){nuli1++;}
                  }
                  printf("zavrsi prebaruvanjeto na deteto %d i najde %d", i,
nuli1);
                  FILE *fp e fopen("com.txt", "a+");
                  if(fp == NULL){
                        printf("Error!");
                        return 0;
                  }
                  fprintf(fp, "%d ", nuli1);
                  fclose(fp);
            }
            else{
                  for(i = 0; i < 10; i++){
                        FILE *fp = fopen("com.txt", "r");
                        if(fp == NULL){
                              printf("Error");
                              return 0;
                        }
                        for(j = 0; j < 10; j++){
                              fscanf(fp, "%d", &nuli1);
                              vkupno1+=nuli1;
                        }
                        fclose(fp);
```

```
                        printf("VKUPNO PRONAJDENI NULI VO DETE PROCESITE:
%d", vkupno1);
                  }
            }
      }
      else{//roditel proces
            deteNitki=fork();//dete so nitki
            if(deteNitki == 0){//vo deteto sme
                  for(j = 0; j < 10; j++){
                        thread_data[j]=j;
                        pthread_create(&nitki[j],  NULL,  prebaraj,  (void
*)&[thread_data]);
                  }
                  for(j = 0; j < 10; j++){
                        pthread_join(nitki[j], NULL);
                  }
                  for(j = 0; j < 10; j++){
                        vkupno2+=nulivonitki[j];
                  }
                  printf("Jas sum dete vo nitki i ima vkupno %d nuli\n",
vkupno2);
            }
            else{
                  wait(NULL) == deteProcesi ? printf("Prvo zavrsi dete so
procesi\n") : printf("Prvo zavrsi dete so nitki\n");
                  wait(NULL);
                  printf("Roditel  doceka  dvete  deca  da  zavrsat  i  toj
zavrsuva.\n");
            }
      }
      return 0;
}
```

Задача5

Да се напише програма во C која работи со процеси и нитки. Главната програма (главната нитка) како аргумент добива име на влезна датотека. Главната нитка треба да креира онолку нитки колку што треба, така што, секоја нитка да добие по 10 линии од влезната датотека (нема повеќе од 1000 линии, а притоа последната нитка може да добие и помалку од 10 линии).Секоја една од нитките, ги изминува своите 10 линии од датотеката и брои колку има големи а колку мали букви. Откако ќе завршат нитките, главната нитка печати на екран колку секоја нитка нашла големи и мали букви и печати колку вкупно големи и мали букви биле пронајдени. Не е дозволено содржината на целата датотека да биде прочитана во низа т.е. секоја од нитките мора да работи со FILE * покажувач за изминување на датотеката т.е. на линиите од датотеката.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <sys/wait.h>
```

```
#include <string.h>

int golemi[100];
int mali[100];
char *filename;

void *prebaraj(void * x){
	int t = *((int *)x);
	int from = t*10;
	int to = t*10+10;
	int lines = 0;
	char c = '';
	FILE *fp = fopen(filename, "r");
	if(fp == NULL){printf("Error");}
	while((c=fgetc(fp))!=EOF){
		if(c=='\n'){
			lines++;
			if(lines>=from && lines<=to){
				while((c=fgetc(fp))!=EOF && c!='\n'){
					if(isalpha(c)){
						if(isupper(c)){golemi[t]++;}
						else{mali[t]++;}
					}
				}
			}
		}
	}
	pthread_exit(NULL);
}

int main(int argc, char *argv[]){
	char *imefile = argv[1];//go zemame fileto
	FILE *file = fopen(imefile, "r");//ja otvorae datotekata
	char c = "";
	int linii=0;
	int i;

	if(argc < 2){
		printf("Premalku argumenti!");
		return 0;

	}
	if(file == NULL){
		printf("Greska vo otvoranjeto na datotekata!");
		return 1;
	}

		while(c != EOF){
			c = fgetc(file);
			if(c=='\n'){
```

```
                        linii++;
                }
        }
        fclose(file);
        int n, brnitki;
        n = linii%10;
        if(n==0){
                brnitki = linii/10;
        }
         else{
                brnitki = linii/10+1;
        }
        pthread_t nitki[brnitki];
        int t[brnitki];
        for(i = 0; i <brnitki; i++){
                t[i]=i;
                pthread_create(&nitki[i], NULL, prebaraj, (void *)&t[i]);
        }
        for(i = 0; i < brnitki; i++){
                pthread_join(nitki[i], NULL);
        }
        int vkm=0;
        int vkg=0;
        for(i = 0; i < brnitki; i++){
        vkm+=mali[i];
        vkg+=golemi[i];
        }

    printf("Mali: %d, golemi: %d", vkm, vkg);
    return 0;
}
```

Задача6

Да се напише C програма која како аргумент добива наредба (без аргументи) што треба да се изврши. Татко процесот креира дете процес кое што ја извршува наредбата, притоа целиот излез од наредбата се запишува во датотека со име „vlez.txt“. Татко процесот чека дете процесот да заврши, а потоа креира 6 нитки кои што ќе ја кодираат содржината на датотеката „vlez.txt“. Најпрво, пред да ги креира нитките, татко процесот ја исчитува содржината на датотеката и ја сместува во низа од најмногу 100 знаци. Потоа, секој еден од процесите изминува по 1/6 од датотеката (првиот првата 1/6, вториот втората и т.н.) и ја кодира соодветната содржина преку повик на функцијата „kodiraj“. Кодирањето се прави така што на секоја буква од низата (на позиција „i“) се додава разликата до следниот знак во низата (на позиција „i+1“). Последниот знак останува непроменет. Откако сите нитки ќе завршат со кодирањето, татко процесот на екран го печати кодираниот текст.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
```

```
#include <unistd.h>

char input[100];//nizata vo koja ke se smesti iscitanata datoteka

void *kodiraj(void *x){
	int i = *((int *)x);
	int start = (100/6)*i;
	int end = (100/6)*(i+1);
	for(i = start; i < end - 1 && i < 100; i++){
		input[i]+=input[i+1];
	}
	pthread_exit(NULL);
}

int main(int argc, char *argv[]){
	int i;
	if(argc < 2){
		printf("Nedovolen broj argumenti!");
		return 1;
	}
	FILE *file = fopen("vlez.txt", "w");
	if(file == NULL){
		printf("Greska pri otvoranje na datotekata!");
		return 1;
	}
	pid_t dete;
	dete = fork();
	if(dete == 0){//vo deteto sme!
		//izlezot od naredbata treba da se zapise vo datotekata!!!???
		execlp(argv[1], argv[1], NULL);
		printf("Neuspesen povik!");
	}
	else{//roditel proces
		wait(NULL);//cekame deteto da zavrsi
		fclose(file);//ja zatvorame datotekata
		//kreirame 6 nitki i ja iscituvame sodrzinata na datotekata vo
niza
		FILE *read_file = fopen("vlez.txt", "r");
		if(read_file == NULL){
			printf("Greska vo otvoranje!");
			return 1;
		}
		fgets(input, 100, read_file);
		fclose(read_file);

		//kreirame 6 nitki
		pthread_t nitki[6];
		int thread_numbers[6];

		for(i = 0; i < 6; i++){
```

```
                  thread_numbers[i]==i;
                  pthread_create(&nitki[i],     NULL,     kodiraj,     (void
*)&thread_numbers[i]);
            }
            for(i = 0; i < 6; i++){
                  pthread_join(nitki[i], NULL);
            }
            printf("Kodiran tekst: %s\n", input);
      }
      return 0;
}
```

Задача7

Да се напише С програма која како аргументи добива непознат број на имиња на текстуални датотеки сместени во тековниот директориум. Родител процесот треба да креира онолку деца процеси колку што има датотеки наведено како аргументи. Секое едно од деца процесите треба да ја пребара соодветно својата датотека во целиот директориум и локациите на кои што ќе ја најде соодветната датотека да ги запише во излезна датотека со исто име како и таа што ја пребарува само со наставка „_izlez“ (првото дете процес ја пребарува првата датотека, второто дете процес втората датотека и т.н.). Откако „i-тото“ дете процес ќе ја пребара „i-тата“ датотека, ќе креира излезна датотека и ќе заврши со извршување, родител процесот креира ново „i-то“ дете процес кое што ќе изврши наредба која што на екран ќе испише колку такви датотеки има пронајдено во системот.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>


void prebaraj_i_zapisi(const char *filename){
      FILE *outputfile;
      char outputfilename[256];

      outputfile = fopen(outputfilename, "w");
      if(outputfile == NULL){
            printf("Greska pri otvoranje!");
            return 1;
      }
/////////direktorium prebaruvanje????!!!!


int main(int argc, char *argv[]){
      pid_t dete;
```

```
	if(argc < 2){
		printf("Nedovolen broj na argumenti");
		return 1;
	}
	for(int i = 0; i < argc - 1; i++){
		dete = fork();
		if(dete == 0){
			prebaraj_i_zapisi(argv[i]);
		}
		else{
			wait(NULL);
			printf("Zavrsija decata!");
		}
	}
	return 0;
}
```

Задача8

Потребно е да се креира C програма која ќе креира 3 нитки (покрај главната нитка). Главната нитка треба да дозволи внесување на N цели броеви од тастатура (не повеќе од 1000), каде што N исто така се внесува од тастатура. Откако ќе бидат внесени целите броеви во низата, главната нитка креира две нитки, каде што првата нитка ги подредува (сортира) позитивните броеви (вклучувајќи ја и нулата) во сопствена низа од броеви, додека пак втората нитка ги подредува (сортира) негативните елементи. Откако двете нитки ќе завршат, главната нитка креира трета нитка која што ги спојува двете низи во една низа преку копирање на соодветните елементи, така што резултантната низа да биде сортирана. Главната нитка ги чека сите 3 нитки да завршат а потоа на екран ја печати содржината на резултантната низа.

```
int niza[1000];
int nizapoz[1000];
int nizaneg[1000];
int rezultantna[1000];
int N;

void *sortpoz(void* x){
	int *data = (int*)x;
	int i,j, temp;
	int index = 0;

	for(i = 0; i < N; i++){
		if(data[i]>=0){
			nizapoz[index++]=data[i];
		}
	}
	//sortiranje
	for(i = 0; i < index-1; i++){
		for(j = 0; j < index-i-1; j++){
			if(nizapoz[i]>nizapoz[j+1]){
```

```
                        temp = nizapoz[i];
                        nizapoz[i] = nizapoz[j+1];
                        nizapoz[j+1] = temp;
                  }
            }
      }
      pthread_exit(NULL);
}

void *sortneg(void* x){
      int *data = int *x;
      int i, j, pom;
      int index = 0;
      for(i = 0; i < N; i++){
            if (data[i] < 0){
                  nizaneg[index++] = data[i];
            }
      }
      for(i = 0; i < index - 1; i++){
            for(j = 0; j < index-i-1; j++){
                  if(nizaneg[i]<nizaneg[j+1]){
                        pom = nizaneg[i];
                        nizaneg[i] = nizaneg[j+1];
                        nizaneg[j+1] = pom;
                  }
            }
      }
      pthread_exit(NULL);
}

void *mesaj(void* x){
      int pindex = 0;
      int nindex = 0;
      int rindex = 0;

      while(pindex < N && nindex < N){
            if(nizpoz[pindex] <= nizaneg[nindex]){
                  rezultantna[rindex++] = pozniza[pindex++];
            }
            else{
                  rezultantna[rindex++] = negniza[nindex++];
            }
      }
      while(pindex < N){
            rezultanta[rindex++] = pozniza[pindex++];
      }

      while(nindex < N){
            rezultanta[rindex++] = negniza[nindex++];
      }
```

```
pthread_exit(NULL);
}


int main(){
	int i;
	printf("Vnesete go N (N<=1000)\n");
	scanf("%d", &N);
	printf("Vnesete %d celi broevi: \n", N);
	for(i = 0; i < N; i++){
		scanf("%d", &niza[i]);
	}

	pthread_t nitkapoz, nitkaneg, nitkarez;

	pthread_create(&nitkapoz, NULL, sortpoz, niza);
	pthread_create(&nitkaneg, NULL, sortneg, niza);

	pthread_join(nitkapoz, NULL);
	pthread_join(nitkaneg, NULL);

	pthread_create(&nitkarez, NULL, mesaj, NULL);
	pthread_join(nitkarez, NULL);

	printf("Rezultantna niza: ");
	for(i = 0; i < N; i++){
		printf("%d", rezultantna[i]);
	}
	printf("\n");
	return 0;
}
```

Задача9

Да се напише програма во C која работи со процеси и нитки. Програмата (главната нитка) како аргументи од командна линија добива збор што се пребарува и листа од имиња на датотеки. Доколку не добие листа од датотеки од командна линија (добие 120 мин. само збор за пребарување), програмата треба да му дозволи да внесе имиња на 3 датотеки од тастатура. Доколку нема ни збор за пребарување, најпрво го внесува зборот од тастатура, а потоа имињата на 3те датотеки. За секоја датотека во која се прави пребарување, се креира посебна нитка која го прави пребарувањето, притоа, на секоја нитка ѝ се испраќа кој е зборот што треба да го пребарува и името на датотеката во која треба да го прави пребарувањето. Секоја нитка го пребарува зборот во својата датотека, и штом заврши, на екран го печати името на датотеката и колку пати се појавил зборот. Откако ќе завршат сите нитки, главната нитка на екран печати, за секоја датотека одделно, колку процентуално се појавил зборот во таа датотека (процент од вкупното појавување на зборот во сите датотеки).

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

```
#include <time.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#define NUM_THREADS 10
typedef struct {
char zbor[20];
char dat[20];
int i;
}info;
int ret[100];
void *funk(void *t){
	info x = *((info *)t);
	FILE *f = fopen(x.dat,"r");
	if(f==NULL){
		printf("ne mozam da ja otvoram datotekata\n");
	}
	int iw=0,z=0;
	char c,zbor[20];

	ret[x.i]=0;
	while((c=fgetc(f))!=EOF){
		if(isalpha(c)){
			if(iw==0){iw=1;}
			zbor[z++]=c;
		}
		else {
			if(iw){
				iw=0;
				zbor[z]=0;
				z=0;
				if(strcmp(zbor,x.zbor)==0){
					ret[x.i]++;
				}
			}
		}
	}
	fclose(f);
	pthread_exit(NULL);
}

int main(int argc,char *argv[]){
	pthread_t nitki[100];
	info lista[100];
	int N,i;
	char zbor[20];
	char datoteki[3][20];
	if(argc>2){
		N = argc-2;
		for(i=0;i<N;i++){
```

```
            strcpy(lista[i].zbor,argv[1]);
            strcpy(lista[i].dat,argv[i+2]);
            lista[i].i=i;
            pthread_create(&nitki[i],NULL,funk,(void *)&lista[i]);
            }
      float vkupno=0.0;
      for(i=0;i<N;i++){
            pthread_join(nitki[i],NULL);
            vkupno+=ret[i];
      }
      for(i=0;i<N;i++){
            printf("Datotekata     %s     e     najdena     %f     procenti
\n",argv[i+2],(ret[i]/vkupno)*100);
      }
      }else {
            if(argc==1){
                  printf("Vnesi go zborot za prebaruvanje\n");
                  scanf("%s",zbor);
            }else {
                  strcpy(zbor,argv[1]);
            }
            printf("Vnesi tri datoteki\n");
            scanf("%s%s%s",datoteki[0],datoteki[i],datoteki[2]);
            N=3;
            for(i=0;i<N;i++){
                  strcpy(lista[i].zbor,zbor);
                  strcpy(lista[i].dat,datoteki[i]);
                  lista[i].i=i;
                  pthread_create(&nitki[i],NULL,funk,(void *)&lista[i]);
            }
            float vkupno=0.0;
            for(i=0;i<N;i++){
                  pthread_join(nitki[i],NULL);
                  vkupno+=ret[i];
            }
            for(i=0;i<N;i++){
                  printf("Datotekata    %s    e    najdena    %f    procenti
\n",argv[i+2],(ret[i]/vkupno)*100);
            }
      }
      return 0;
}
```