ZADAĆA 1: DINAMIČKA ALOKACIJA MEMORIJE

Svaki zadatak nosi određeni broj bodova po podzadatku koji su naznačeni u zagradi. Prvi zadatak treba biti predan kao txt dokument (napravljen u SC5), a ostali kao C++ programski kod. Svi student koji predaju zadaće trebaju pristupiti na ScriptRunner5 sustav (https://mathos.scriptrunner.carnet.hr/) I tamo objavljivati svoje zadaće u mapi ZADACE->ZADACA_1.

Svi oni student koji predaju zadaće moraju biti nazočni na vježbama.

ZADATAK 1 (5+5+5):

Sljedeća pitanja se tiču razumijevanja korištenja pokazivača i dinamičke alokacije memorije u C++ programskom jeziku. Za svaki odgovor dajte objašnjenje.

1. Koji od sljedećih naredbi je točan način dinamičke alokacije varijable tipa int? Objasnite svojim riječima.

   ☐ `int* x = new int;`

   ☐ `int x = new *int;`

   ☐ `int *x = int new;`

2. Što je problem u sljedećem programskom kodu? Objasnite svojim riječima.

```
int main()
{
    int A[]={1,3,1,12,34,5};

    int *p = A;

    delete [] p;
}
```

3. Koje sve probleme vidite u sljedećem programskom kodu? Objasnite svojim riječima.

```
void memcpy(float *x, float *y, int n)
{
	y = new int[n];
	for(int i=0;i<n;i++) y[i]=x[i];
}

int main()
{
	int A[]={3,4,1,5,7,9,10};
	int *y;
	memcpy(x,y,sizeof(A)/sizeof(int));

	for(int i=0;i<n;i++) cout << y[i] << " ";
}
```

ZADATAK 2 (10+5+5):

Neka su dani sljedeći potpisi funkcija

```
void *transpose(float *A, int m, int n);

void ispis(float *A, int m, int n);
```

Učinite sljedeće:

1. U proceduri `transpose` implementirajte transponiranje matrice tako da za matricu A koje je spremljeno kao **jednodimenzionalno polje** veličine mn učinite transponiranje te matrice. Nakon poziva `transpose(A)` matrica A postaje A^T.
2. Implementirajte proceduru ispis koja će ispisati matricu A u formatiranom obliku od m redaka i n stupaca
3. Testirajte program na sljedećoj matrici:

$$A = \begin{bmatrix} -1 & 2 & 3 & 0 \\ 0 & 2 & 5 & 1 \\ 4 & 3 & 1 & 0 \\ 1 & 2 & -5 & 7 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

Ostali detalji implementacije su proizvoljni.

ZADATAK 3 (20+5)

Neka je dan sljedeći potpis funkcije

```
void matrix_multiply(float *A, float *B, float *C, int m, int k, int n);
```

Učinite sljedeće:

1. U proceduri `matrix_multiply` implementirajte množenje realnih matrica A i B veličine $m \times k$ odnosno $k \times n$ čiji rezultat spremite u $m \times n$ matricu C.
2. Testirajte program na sljedećim matricama:

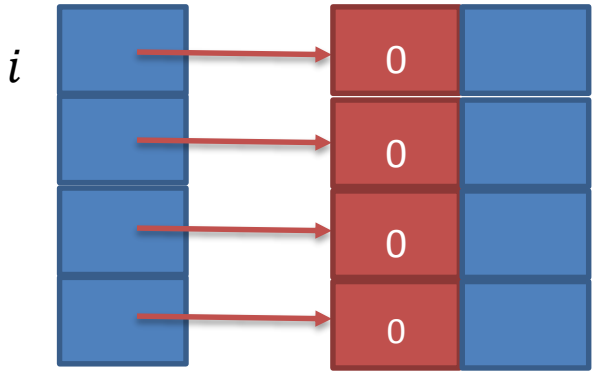
$$A = \begin{bmatrix} -1 & 2 & 3 & 0 \\ 0 & 2 & 5 & 1 \\ 4 & 3 & 1 & 0 \\ 1 & 2 & -5 & 7 \\ 0 & 0 & 2 & 1 \end{bmatrix}, B = \begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & 3 \\ 1 & 3 & 0 \\ 9 & -2 & 4 \end{bmatrix}$$

Ostali detalji implementacije su proizvoljni.

ZADATAK 4 (10 +20+10)

Neka je A struktura podataka koja implementira polje u kojem je svaki element pokazivač na neko polje cijelih brojeva. Pretpostavimo da se nad tom strukturom dodaju vrijednosti operacijom `add_element(A,i,x)` koja na prvo slobodno mjesto retka $A[i]$ dodaje vrijednost x.

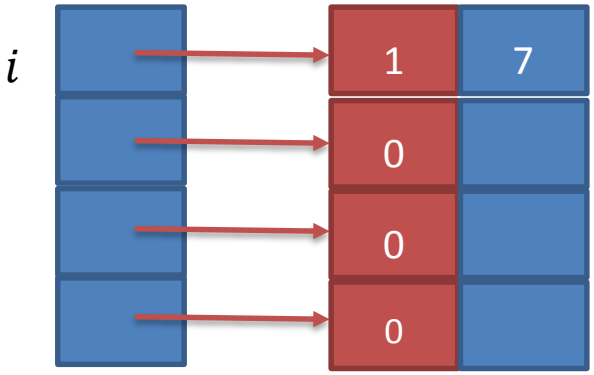
Pretpostavit ćemo da operacije `add_element` dolaze 'on-line', tj. njihov redoslijed i pojava su unaprijed nepoznati, stoga struktura A treba omogućiti efikasno spremanje takvih podataka. Nadalje, pretpostavit ćemo da se u samom početku struktura A sastoji od n pokazivača tipa int te da svaki od tih pokazivača pokazuje na polje duljine 2. Možete pretpostaviti da $0 \leq i < n$.



Neka prvi element polja $A[i]$ sprema trenutni broj njegovih elemenata. Na slici gore vidite inicijalno stanje strukture A za $n = 4$. Nakon operacije

```
add_element(A,0,7)
```

struktura A će izgledati ovako:



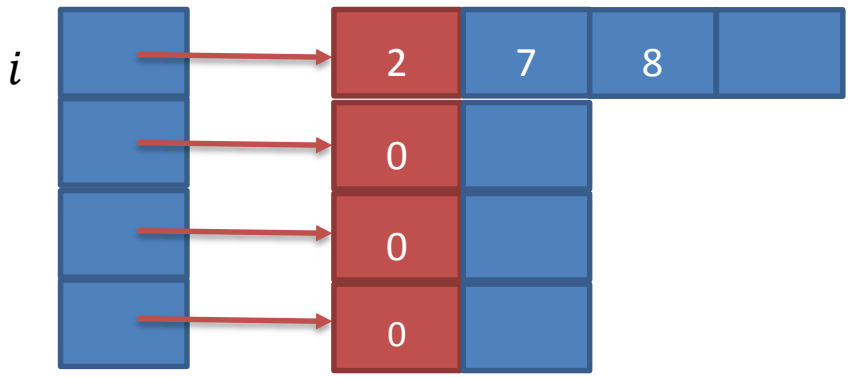
Učinite sljedeće:

1. Implementirajte inicijalnu strukturu A kao što je gore objašnjeno i implementirajte funkciju `add_element` koja dodaje cijeli broj u i-to polje strukture A, ukoliko u i-tom polju ima slobodnog mjesta. U suprotnom ispisuje poruku da je nemoguće dodati element.

2. Implementirajte metodu `dynamic_add(A,i,x)` koja pozove `add_element(A,i,x)` ukoliko ima mjesta za x u polju $A[i]$. Ukoliko nema više mjesta u polju $A[i]$, zamjeni $A[i]$ sa duplo većim poljem i onda pozovi `add_element(A,i,x)`.

   NAPOMENA: uočite da se veličina polja (broj elemenata polja) $A[i]$ može uvijek lako odrediti kao najmanja potencija broja 2 takva da $2^y > A[i][0]$, za neki $y \geq 1$.

   PRIMJER: `dynamic_add(A,0,8)`



3. Napišite funkciju `print_db(A, n)` koja ispisuje strukturu A na „oku prihvatljiv“ način ☺

NAPOMENA: Kod se nalazi na sljedećoj stranici

```
#include<cstdlib>
#include <iostream>
#include<time.h>

using namespace std;

void add_element(int **A, int i, int x){}

void dynamic_add(int **A, int i, int x){}

void print_db(int** A, int n){}


int main()
{
	int** A;
	int n = 20;
	// alocirate ovdje inicijalno stanje strukture A kao sto je
	// opisano u tekstu zadatka.
	/****** ovdje ide vasa implementacija ******/


	/****** ovdje zavrsava vasa implementacija ******/
	print_db(A,n); // ispisi inicijalno stanje na ekran

	// Simulirajte online-algoritam na nacin da generirate
	// 'size' slucajnih cijelih brojeva i indeksa redaka 'i' od 'A[i]'.
	// U nastavku vam je dan kod za to.

	srand(time(0));
	int size = 100;
	for(int k=0;k<size;k++)
	{
	  int i = rand()%n;//generiranje slucajnog broja u {0,1,...,n-1}
	  int x = rand()%100+1;//generiranje slucajnog broj u {1,2,...,100}
	  dynamic_add(A, i, x);
	}

	print_db(A,n); // ispisi stanje nakon 'size' mnogo dodavanja.

}
```