November 15, 2021

CS 204 (Advanced Programming) MIDTERM I

1	2	3	4	5	6	TOTAL

Name and Last Name :
ID :
SUNet Username :

Notes:
a) Duration is 90 minutes. Total number of points is 105 (the extra 5 points are bonus☺)
b) Closed-book, closed-notes, no calculators and computers. A single sided A4 size handwritten cheat-note page is allowed.
c) There must be six pages (including this one) in this booklet. Please check it out!
d) Write your name at the designated area of each page

QUESTIONS

1) (16 points in total)
What is the output of the following program?

```
#include <iostream>
using namespace std;

void function (int *& refPtr, int * ptr)
{
	cout << "function 1: " << ++(*refPtr) << " " << *ptr << endl;
	delete refPtr;
	refPtr = new int(10);
	*ptr = 11;
	cout << "function 2: " << *refPtr << " " << *ptr << " " << endl;

}
int main()
{
	int * ptrA = new int;
	int b = 5;
	int &c = b;
	*ptrA = 7;
	c = 1;
	cout << "main 1: " << *ptrA << " " << b << endl;
	function(ptrA, &b);
	cout << "main 2: " << *ptrA << " " << c << endl;
	return 0;
}
```

```
main 1: 7 1
function 1: 8 1
function 2: 10 11
main 2: 10 11
```

Açıklamalı [WK1]: Each line has 4 points.
Each variable output of each line has 2 points

NAME:

2) (15 points in total) Consider the following main program and `myCode.cpp` file.

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
	#define _CS201 "cs204"
	string myCourse = _CS201;
	#include "myCode.cpp"
	cout << myCourse << endl;

	#ifndef _CS204
		cout << "CS204 is printed" << endl;
	#else
		cout << "CS204 is not printed"<< endl;
	#endif

	if (OPER * 2 > 6)
		cout << "this line is printed" << endl;
	else
		cout << "this line is not printed" << endl;
	return 0;
}
```

myCode.cpp

```
#ifdef _CS201
myCourse = myCourse+" is easy";
#endif
#define _CS204
#define OPER 3+1
```

a) (11 points) What is the translation unit that corresponds to the main function? Fill in the box below!

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
	string myCourse = "cs204";
	myCourse = myCourse+" is easy";
	cout << myCourse << endl;
	cout<<"CS204 is not printed"<<endl;

	if (3+1 * 2 > 11)
		cout << "this line is printed" << endl;
	else
		cout << "this line is not printed" << endl;

	return 0;
}
```

Açıklamalı [WK2]: Each line has 1.25 points
If all are correct 11 points are given

b) (4 points) Does this program compile and link correctly? If not, specify the erroneous lines in the translation unit. If so, what is the output?

The programs compiles and runs without any problem. The output is:

```
cs204 is easy
CS204 is not printed
this line is not printed
```

Açıklamalı [WK3]: Each line has 1 point (1 point for the explanation and 1 point for each output line)

NAME:

3) (15 points in total)

a) (3 points) Using the new command dynamically allocate an array of 100 integers and assigned it to an integer pointer named `myArr`!

```
int * myArr = new int[100];
```

Açıklamalı [WK4]: int * myArr is 1.5 points,
new int [100] is 1.5 points

b) (3 points) Using the `calloc` command dynamically allocate an array of 100 integers and assigned it to an integer pointer named `myArr`!

```
int * myArr = (int *) calloc (100, sizeof (int));
```

Açıklamalı [WK5]: int * myArr is 1 point,
casting (int *) is 1 point,
calloc (100, sizeof (int)) is 1 point

c) (3 points) Using the `malloc` command dynamically allocate an array of 100 integers and assigned it to an integer pointer named `myArr`!

```
int * myArr = (int *) malloc (100 * sizeof (int));
```

Açıklamalı [WK6]: int * myArr is 1 point,
casting (int *) is 1 point,
malloc (100*sizeof (int)) is 1 point

d) (3 points) An integer pointer named `myArr` points to a dynamically allocated array of size 100 (it could have been allocated by either `new`, `calloc` or `malloc`). Write a single command that will delete the dynamically allocated array to which `myArr` is pointing to from the heap!

```
delete [] myArr;
or
free(myArr);
```

Açıklamalı [WK7]: if [ ] are forgotten for delete, only 1 point is given

e) (3 points) You are running your code in debug mode. Just before dividing integers a and b and assigning the result to another integer c, in order to avoid the program crashing, write an assert statement that will make the program to abort if b is zero! Does this code work in Release mode?

```
assert(b!=0);
c=a/b;
It does not work in Release mode
```

Açıklamalı [WK8]: if c=a/b is not written, no penaltiy is given
2 points for assert(b!=0)
if assert(b==0) is written no points are given

Açıklamalı [WK9]: 1 point

NAME:

4) (20 points in total) The node structure of a doubly linked list is given below. You should write a function that inserts a new node into **an ascendingly sorted doubly linked list** and keeps the list sorted after the insertion is done. The prototype of the function is:

```
node * addANode(node* head, int newInfo);
```

where head is the head pointer of an ascendingly sorted doubly linked list and newInfo is the info value of the new node that should be added into the doubly linked list. The function returns the head of the doubly linked list after the insertion is done.

Hint: you should take into consideration the cases when the head input parameter is NULL, when you should insert the new node as the first one in the list, somewhere in the middle of the list, as the last one of the list, etc. For your convenience we advise you to use the given node constructor.

Note: In this question, you may prefer not to attempt to solve it by signing the "not attempted" box below and secure 4 points. If you sign the "not attempted" box below, you accept that you did not answer this question and you will receive 4 points. In this case, your answer will not be graded even if you write something as solution

NOT ATTEMPTED!

```
struct node
{
	int info;
	node * next;
	node * prev;
	node(int i, node * n, node * p): info(i), next(n), prev(p)
	{}
};

node * addANode(node* head, int newInfo)
{
	node * ptr = head;
	node * temp;
	if(head == nullptr || newInfo < head->info)
	 //we have a new head which is pointing to the new node
	{
		temp = new node(newInfo, head, nullptr);
		if(head!=nullptr)
			head->prev = temp;
		return temp; //the new head is returned, the f-ion is finished
	}
	//otherwise no new head and we find the place for the new node
	while(ptr->next!=nullptr && ptr->next->info < newInfo)
		ptr = ptr->next;
	temp = new node(newInfo, ptr->next, ptr);
	if(ptr->next!=nullptr) //if the new node is not at the end of the list
	{
		temp->next->prev = temp;
	}
	ptr->next = temp;
	return head; //return the old head
}
```

Açıklamalı [WK10]: 6 points if the new node is correctly inserted at the beginning of the list and the new head is returned

Açıklamalı [WK11]: 7 points are awarded if the node is correctly inserted in the middle and 7 points are awarded if the node is correctly inserted at the end of the list and the old head is returned

NAME:

5) (14 points in total) Write a function that as a parameter takes **a reference to a stack**, prints the content of this stack and at the end of the function the state (content) of the stack is the same as it was before the start of the function.

Note: the member functions of the stack class that you might need in this question are: `isEmpty()`, `push(int)` and `pop(int &)`. You should know what they are doing.

```
void printAStack(DynIntStack & theStack)
{
      int tempInt;
      DynIntStack tempStack;

      while(!theStack.isEmpty())
      {
            theStack.pop(tempInt);
            cout<<tempInt<<endl;
            tempStack.push(tempInt);
      }
      while(!tempStack.isEmpty())
      {
            tempStack.pop(tempInt);
            theStack.push(tempInt);
      }
}
```

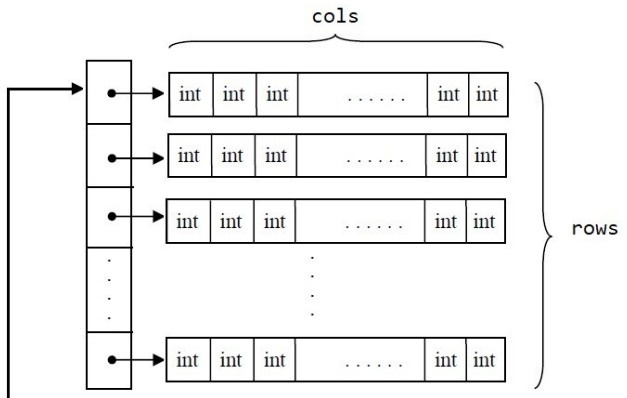
Açıklamalı [WK12]: 2 points for non-const stack reference

Açıklamalı [WK13]: 5 points for printing the stack

Açıklamalı [WK14]: 7 points for restoring the stack at the end

NAME:

6) (25 points in total). Below you have the definition/illustration of the class `DynIntMatrix`. As its private data members it has a pointer to pointer to integer named `myMatrix` which points to a dynamically allocated matrix (2D array), as well as the integers `rows` and `cols` which store the number of rows and columns, respectively, of the dynamically allocated matrix to which `myMatrix` is pointing to.




```
class DynIntMatrix
private:
    int ** myMatrix;
    int rows;
    int cols;

public:
    ...
    DynIntMatrix(const DynIntMatrix & copyMatrix);
    ~DynIntMatrix();
    ...
```

a) (5 points) Write a shallow copy constructor for the class `DynIntMatrix`!

```
DynIntMatrix::DynIntMatrix(const DynIntMatrix & copyMatrix)
{
    myMatrix = copyMatrix.myMatrix;
    rows = copyMatrix.rows;
    cols = copyMatrix.cols;
}
```

Açıklamalı [WK15]: 2 points for taking a reference parameter and correctly writing the constructor head

Açıklamalı [WK16]: 3 points for each initialization line in the shallow copy constructor body

NAME:

b) (12 points) Write a deep copy constructor for the class `DynIntMatrix`!

```
DynIntMatrix::DynIntMatrix(const DynIntMatrix & copyMatrix)
{
	rows = copyMatrix.rows;
	cols = copyMatrix.cols;

	myMatrix = new int*[rows];
	for(int i = 0; i < rows; i++)
	{
		myMatrix[i] = new int[cols];
	}

	for(int i = 0; i < rows; i++)
	{
		for(int j = 0; i < cols; j++)
		{
			myMatrix[i][j] = copyMatrix.myMatrix[i][j];
		}
	}
}
```

Açıklamalı [WK17]: 2 points, one for each initialization

Açıklamalı [WK18]: 2 points for dynamically allocating memory for myMatrix, 2 points for dynamically allocating memory for each element (row) of the myMatrix and 2 points for the order by which these dynamic allocations are done

Açıklamalı [WK19]: 4 points for initializing the values of the this->matrix to be as in the copyMatrix.myMatrix.

c) (8 points) Write the destructor of the class `DynIntMatrix`!

```
DynIntMatrix::~DynIntMatrix()
{
	for(int i = 0; i < rows; i++)
	{
		delete [] myMatrix[i];
	}
	delete [] myMatrix;
}
```

Açıklamalı [WK20]: 3 points for deleting the rows of the matrix, 2 points for deleting the myMatrix column, and 3 points for the order by which they are done