March 25, 2013

CS204 (Adv. Prog.) First Midterm Exam

1	2	3	4	5	6	7	TOTAL

Name and Lastname : Özlem Listebağlar
SUNET ID :

Notes: a) Please answer the questions only in the provided space after each question.
b) Duration is 100 minutes.
c) Closed-book, closed-notes, no calculators and computers. ½ single sided A4 size handwritten cheat-note page is allowed.
d) There must be seven pages (including this one) in this booklet. Please check it out!

QUESTIONS

1) (10 points) What is the output of the following program?

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

int  myFunc (int ** a)
{
    **a = 7;
    *a = new int;
    **a = 12;
    return (**a + 3);
}

int * theFunc (int * a)
{
    static int x = 4;
    *a = x;
    x++;
    return new int;
}

int main ()
{
    int * p = new int;
    int * q;
    *p = 6;
    q = theFunc(p);
    cout << "main1 " << *p <<  endl;
    int * r = theFunc(q);
    cout << "main2 " << *p << " " << *q << endl;
    int * t = p;
    *r = myFunc(&p);
    cout << "main3 " << *p << " " << *q << " " << *r << " " << *t << endl;
}
```

```
main1 4
main2 4 5
main3 12 5 15 7
```

NAME:

2) (12 points) Consider the following *main.cpp* and *others.cpp* files which are part of the same project in VS2010.

main.cpp

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

#define FOUR 2*3-2

void change_it (int num);

extern int glob;

int main()
{
    #if (FOUR*FOUR == 16)
        cout << "Output is: ";
    #elif (FOUR*FOUR == 8)
        cout << "Value is: ";
    #elif (FOUR*FOUR == 0)
        cout << "Sonuc: ";
    #endif
    glob = 24;
    cout << glob << endl;
    change_it(2);
    cout << glob << endl;

    return 0;
}
```

others.cpp

```
int glob = 9;

void change_it(int num)
{
    glob = glob + num;
    #ifdef FOUR
        glob++;
    #endif
}
```

a) Cross the lines which will **not** be included in the translation units in above files.

b) Do these two files compile correctly? If not, specify the erroneous lines in the file(s) and do not continue with part (c). If so, give the number of object files created and continue with part (c).

2 object files are created; one is for *main* and the other is for *others*

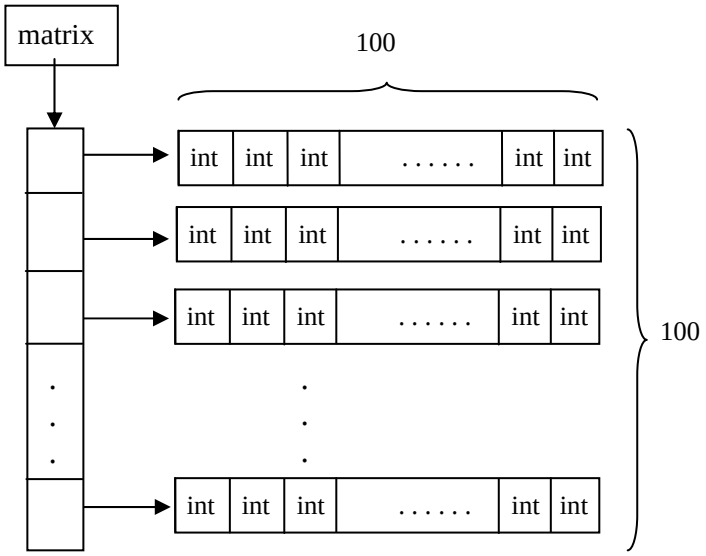
c) Do these codes link correctly? If not explain why. If so, give the output of the program.

Yes they link correctly. Output is:

```
24
26
```

NAME:

3) (15 points) Consider the following dynamic 2D square array structure with 100 rows and 100 columns. The box named *matrix* is an appropriate pointer to reach the matrix elements. This structure is **not** implemented as a class.



The function below is a partial solution for the following problem:

> Write a function that takes such a *matrix* as parameter and returns the sum of main diagonal (from top-left to bottom right) and the sum of antidiagonal (from top-right to bottom-left) as two different parameters.

However, the function is incomplete. Complete this function by filling in the boxes with appropriate parameters at the function heading and appropriate expressions in the function body.

You are not allowed to delete or update anything. Moreover, you cannot add anything other than the code that you are going to write in the boxes.

Hint: In this current form of these boxes, `[ ]` notation cannot be used to reach matrix elements; try pointer arithmetic.

```
void sumDiagonals ( int ** matrix , int * sumMain , int * sumAnti )
{
      int * row;
      int i;
      *sumMain = 0;
      *sumAnti = 0;
      for (i=0; i<100; i++)
      {
            row = * ( matrix + i );
            *sumMain += * ( row + i );
            *sumAnti += * ( row + 99 - i );
      }
}
```

In this question, you may prefer not to attempt to solve it by signing the "not attempted" box below and secure 3 points. If you sign the "not attempted" box below, you accept that you did not answer this question and you will receive 3 points. In this case, your answer will not be graded even if you write something as solution.

Not attempted

NAME:

4)

a) (2 points) Write the necessary piece of code to display `Real Madrid is not real` only in **debug** configuration.

```
#ifdef _DEBUG
     cout << "Real Maidrid is not real";
#endif
```

b) (3 points) Write the declaration for a global string variable which can be accessed only in the cpp file in which it is declared, not in another cpp file of the same project. Name this variable as *className* and initialize to "CS204 is great".

```
static string className = "CS204 is great";
```

c) (6 points) Consider the following declarations:

```
double * doubarray = (double *) malloc (400);
long * longarray = new long [29];
int intarray [80];
```

(i) Fill in the box below with an appropriate C++ expression so that the output gives the amount of bytes allocated from the runtime stack part of the memory for the abovementioned declarations.

```
cout << [ 80*sizeof(int)+sizeof(double *)+sizeof(long *) ] <<
        " bytes have been used from runtime stack" << endl;
```

(ii) Fill in the box below with an appropriate C++ expression so that the output gives the amount of bytes allocated from the heap memory for the abovementioned declarations.

```
cout << [ 400+29*sizeof(long) ] <<
        " bytes have been used from the heap memory" << endl;
```

NAME:

5) (18 points) Consider the following regular linked list `node` struct definition. Please remark that there is no constructor defined for this struct (i.e. you have to use only without a constructor).

```
struct node {
      double info;
      node *next;
};
```

Write a function that takes the head pointers of two such linked lists as parameters, say `myList1` and `myList2`, and generates a third linked list, say `myList3`, which is made up by the nodes with `info` fields are the sum of the `info` fields of the corresponding nodes of `myList1` and `myList2`. In other words, the first node of `myList3` contains `info` that is the sum of the `info` fields of the first nodes of `myList1` and `myList2`; similarly second node of `myList3` contains the sum of the second nodes of `myList1` and `myList2`; and this goes on. The function should return the head pointer of `myList3` as the function's return value.

It is assumed that there are equal number of nodes in `myList1` and `myList2`. However, both lists may be empty.

Linked list is **not** implemented as a class.

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

```
node * addLists (node * myList1, node * myList2)
{
      node * p1 = myList1;
      node * p2 = myList2;
      node * p3;
      node * myList3 = NULL;
      if (p1 != NULL)
      {
            p3 = new node;
            p3->info = p1->info + p2->info;
            p3->next = NULL;
            myList3 = p3;
      }
      else
            return myList3;
      while (p1->next != NULL)
      {
            p3->next = new node;
            p3 = p3->next;
            p1 = p1->next;
            p2 = p2->next;
            p3->info = p1->info + p2->info;
            p3->next = NULL;
      }
      return myList3;
}
```

NAME:

6)

a) (7 points) Consider the following `node` struct definition, which is used to generate a linked list.

```
struct node {
      int value;
      node *next;
};
```

The following incomplete function is to add a new node to the beginning of a linked list pointed by the `head` pointer. The `value` field of this node is given as the second parameter of the function. **<u>This function should not a make the requested operation if there is no enough memory for this new node</u>**.

Complete this function by filling the boxes with appropriate expressions. You are not allowed to delete or update anything. Moreover, you cannot add anything other than the code that you are going to write in the boxes

```
void addToBeginning (node * & head, int num)
{
      node *toBeAdded = new node;

      if ( toBeAdded != NULL )
      {
            toBeAdded -> value = num;

            toBeAdded -> next = head;

            head = toBeAdded ;
      }
}
```

b) (9 points)

Consider you have a struct called `student`. Write a function that that takes a built-in array of `student`, called `myStudents` and a student pointer, called `stuPtr`. Assuming that `stuPtr` points to an element of `myStudents`, return the corresponding array index from the function. Function's heading is given below for your convenience; write down the function body as requested.

```
int index (student myStudents[], student * stuPtr)
```

```
{
      return ((int)stuPtr - (int)myStudents) / sizeof(student);

      return (stuPtr-myStudents);

      int i = 0;
      while (&myStudents[i] != stuPtr)
            i++;
      return i;
}
```

Alternative 1

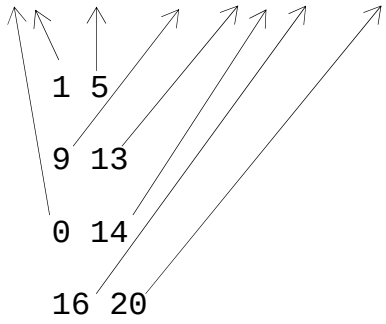
Alternative 2

Alternative 3

NAME:

7) (18 points)
Firstly, let us remember the parenthesis matching rule in a mathematical expression: each close parenthesis ')' belongs to the closest open parenthesis '(' on the left for which there is no close one. For example, consider the following expression and the position (index) pairs of the matching parentheses.

```
((a+b)/4*(e+f))/(2+t)
```



The function below is a partial solution for the following problem:

> Write a function that takes a string for a mathematical expression and displays the index pairs of matching parentheses with the help of a dynamic integer stack. Indexing starts with 0. Assume that the expression is in correct format.

However, the function is incomplete. Complete this function by filling the boxes with appropriate statements.

You are not allowed to delete or update anything. Moreover, you cannot add anything other than the code that you are going to write in the boxes. Furthermore, you are not allowed to change the `DynIntStack` class. In other words, you can use only `DynIntStack` class' `push`, `pop`, and `isEmpty` member functions.

```
void matchParentheses (const string & expr)
{
      DynIntStack myStack;
      int i, j;
      cout << "Indices of the matching parentheses are:" << endl;
      for (i=0; i < expr.length(); i++)
      {
            if ( [ expr[i] == '(' ] )
            {
                  myStack. [ push(i) ] ;
            }
            else if ( [ expr[i] == ')' ] )
            {
                  myStack. [ pop(j) ] ;

                  cout [ << j << " " << i << endl ] ;
            }
      }
}
```

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