March 19, 2016

CS204 (Adv. Prog.) First Midterm Exam

1	2	3	4a	4b	4c	4d	5	6a	6b	7	TOTAL

Name and Lastname : Debug Failure
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 (half) A4 size, single-sided, handwritten cheat-note page is allowed.
d) There must be seven pages (including this one) in this booklet. Please check it out!

QUESTIONS

1) (9 points) What is the output of the following program?

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

int * lotsofptr (int * iptr1, int * iptr2)
{
   *iptr1 = *iptr1 + 3;
   cout << "in func: " << *iptr1 << " " << *iptr2 << endl;
   return  iptr2;
}

int main()
{
   int i;
   int num = 9;
   int * iptr = new int;
   *iptr = 14;
   iptr = lotsofptr(&num, iptr);
   cout << "in main first: " << num << " " << *iptr << endl;
   iptr = lotsofptr (iptr, &num );
   cout << "in main second: " << num << " " << *iptr << endl;

   return 0;
}
```

Write your answer here

```
in func: 12 14
in main first: 12 14
in func: 17 12
in main second: 12 12
```

NAME:

2) (12 points) Consider the given *months.cpp* and *change.cpp* files, which are **part of the same project** in VS2012.

months.cpp

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

string cat;
extern string dog;
void change();

#define MONTH 12
#define TAX

int main()
{
   cat = "march";
   dog = "april";
   change();
   cout << "in months: " << cat << " " << dog<<endl;

   #ifdef MONTH
      cout << 5 - TAX 3 << endl;
   #ifndef TAXMONTH
      cout << MONTH << endl;
   #endif
   #endif
   #if MONTH - 5 == 7
      cout << "gevrek" << endl;
   #else
      cout << "simit" << endl;
   #endif
   #undef MONTH

   return 0;
}
```

change.cpp

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

string dog;
extern string cat;

void change()
{
   cout << "in change: " << cat
        << " " << dog << endl;
   cat = "may";
   dog = "june";
}
```

a) What is the <u>translation unit</u> that corresponds to *months.cpp*? Fill in the box below.

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

```
string cat;
extern string dog;
void change();


int main()
{
   cat = "march";
   dog = "april";
   change();
   cout << "in months: " << cat << " " << dog<<endl;

      cout << 5 -  3 << endl;
      cout << 12 << endl;
      cout << "gevrek" << endl;

   return 0;
}
```

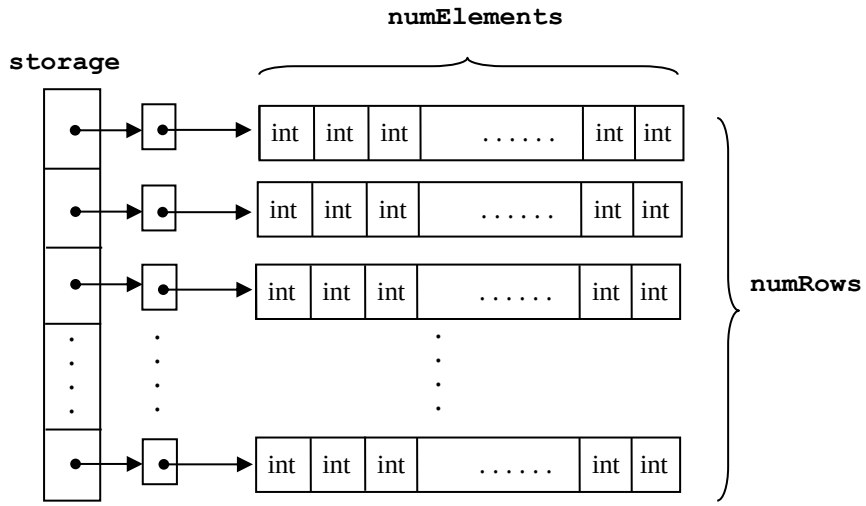
b) Does this project compile and link (i.e. build) correctly? If not, specify the erroneous lines on the code and explain the error. If so, give the output of the program? In either case, write your answer in the box below.

Yes it builds and links correctly.

The output is:

```
in change: march april
in months: may june
2
12
gevrek
```

3) (16 points) Consider the following dynamic 2D storage that contains dynamically allocated `numElements` integers in each row. Each row is pointed by a pointer. The pointer of each row is also pointed by a pointer to pointer, which is an element of a built-in array (called `storage`). The size of this built-in array is the number of rows (`numRows`).



Write a function that takes `storage`, `numRows` and `numElements` as parameters and returns the sum of the values stored in such 2D storage. The header of the function is given below for your convenience. Please fill the body. Assume `numRows` and `numElements` are positive.

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

```
int sumMatrix (int ** storage[], int numRows, int numElements)
{
```

```
int i, j;
int sum = 0;
for (i=0; i < numRows; i++)
{
     int * p = *storage[i];
     for (j=0; j < numElements; j++)
     {
          sum = sum + p[j];
     }
}
return sum;
```

```
}
```

NAME:

4)

a) (4 points) Consider the following macro definition.

```
#define ZERO(A)   A - A
```

What is the output of the following program piece? Show your work.

```
cout << ZERO(12) / ZERO(3) << endl;
cout << (ZERO(12)) * (ZERO(3)) << endl;
```

```
5
0
```

b) (3 points) What is the output of the following program piece?

```
double * darr = new double [25];
if (darr == (&darr[5] - 5))
       cout << "Sampiyon Kartal" << endl;
else
       cout << "Sampiyon Fener" << endl;
```

```
Sampiyon Kartal
```

c) (6 points) Consider the following statements.

```
char * x = (char *) malloc (500);
int * y = (int *) calloc (500,4);
```

Considering that one integer uses four bytes and one char uses one byte of memory, answer the following questions.

- How many **integers** are dynamically allocated?

  500 integers are dynamically allocated

- How many **characters** are dynamically allocated?

  500 characters are dynamically allocated

- How many bytes have been dynamically allocated in total?

  500x4 + 500 = 2500 bytes have been dynamically allocated

d) (2 points) Write the necessary piece of code in order to increment the value of variable `xnum` only in **release** configuration.

```
#ifdef NDEBUG
    xnum++;
#endif
```

NAME:

5) (16 points) Write a function that takes a node pointer, say `thePtr`, as a parameter. `thePtr` points to one of the nodes of a **doubly linked list**. The function that you will implement will delete all of the nodes of this doubly linked list. The node that `thePtr` points may be any node of the list; not necessarily head or tail. The function heading is given below for your convenience. Write the function body within the box.

You do not need to know the entire `node` struct for this question; you just need to know that the next pointer field is named as `next` and the previous pointer field is named as `prev`. Moreover, you do not need to know the head and tail pointers of the list.

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

You may assume that the list is **not empty**. However, there could be any number of elements in it.

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

```
void deleteList (node * thePtr)
{
    node * p, * temp;
    p = thePtr->next;
    while (p != NULL)
    {
          temp = p;
          p = p->next;
          delete temp;
    }
    p = thePtr->prev;
    while (p != NULL)
    {
          temp = p;
          p = p->prev;
          delete temp;
    }
    delete thePtr;
}
```

NAME:

In this part (a) of the question, you may prefer not to attempt to solve it by signing the "not attempted" box below and secure 2 points. If you sign the "not attempted" box below, you accept that you did not answer this question and you will receive 2 points. In this case, your answer will not be graded even if you write something as solution.

Not attempted

6)

a) (8 points) What is the output of the following program piece?

```
int i;
int arr [4];
for (i=0; i<4; i++)
{
      arr[i] = i + 1;
}

int * dyn_arr = new int[4];
int * parr = &arr[0];
int * pdyn_arr = &dyn_arr[3];

while (parr <=  &arr[3])
{
      *pdyn_arr-- = *parr++;
}

parr = &arr[0];
for (i=0; i<4; i++)
{
      cout << *(parr + i) << " " << dyn_arr[i] << endl;
}
```

```
1 4
2 3
3 2
4 1
```

b) (8 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 aims to **return the address of the last element** (i.e. a pointer to the last element) of the linked list. The first element of the list is pointed by the `head` pointer. The list may or may not be empty.

Complete this function by filling in the boxes with appropriate code. 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

```
[ node *          ] tailAddress (node * head)
{
      if ( [ head == NULL          ] )
      {
            return NULL;
      }
      node * ptr = head;

      while ( [ ptr->next != NULL          ] )
      {
            ptr = [ ptr->next          ] ;
      }
      return ptr;
}
```

NAME:

7) (16 points) The function below is a partial solution for the following problem:

> Write a function that takes a `DynIntStack` (say `myStack`) and an integer (say `searchNum`) as parameters. The function should return true if `searchNum` exists in `myStack`; returns false otherwise. You may push/pop to/from `myStack` within the function, but at the end of the function, the values stored in `myStack` should be restored to the original values.

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

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. Remember that `push` and `pop` are void functions and they take integer parameters. `isEmpty` returns Boolean.

```
bool findInStack (DynIntStack & myStack, int searchNum)
{
  int num;
  bool found = false;
  DynIntStack temp;

  while ( [ !myStack.isEmpty() ] && found == false )
  {
    myStack.[ pop(num) ];

    temp.[ push(num) ];

    if ( [ num == searchNum ] )
        found = true;
  }

  while ( [ !temp.isEmpty() ] )
  {
    temp.[ pop(num) ];

    myStack.[ push(num) ];

  }
  return found;
}
```

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