Université d'Ottawa
Faculté de génie

École d'ingénierie
et de technologie de l'information

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

ITI 1120 Winter 2011
Introduction to Computing I
Midterm Exam

Exam Duration: 90 minutes **March 5th, 2011, 12:00-13:30**

Professor: Mohamad Eid

Instructions: Please read carefully!

1. **Complete all sections of the identification area in ink.**
2. This is a closed-book test. **No books, papers, or electronic devices are permitted. Non-programmable calculators are permitted.**
3. Answer all questions on the question sheet in the area provided. **Questions answered in pencil will not be re-graded even if there is a marking error.**
4. The marks allocated to each question are indicated. Not all questions are worth the same amount, so plan your time accordingly. The midterm will be scored out of 40 marks, which represents 20% of your final grade.
5. Algorithms are to be described using the format from the lectures and the notes.
6. You can use the back of the question sheet pages, or page 9, for calculations and other work. Pages 9 and 10 can be detached as they will not be marked.
7. Les réponses en français sont acceptées.

Identification:

Name: _____

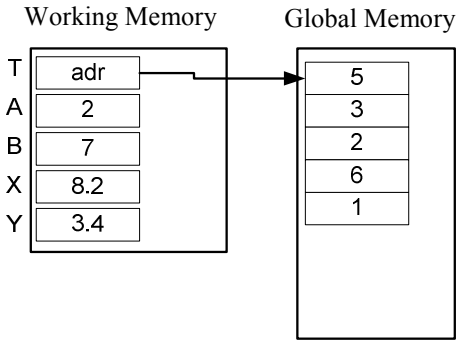
Student number: _____

For use of grader:

Question	Marks available	Marks received
1	8	
2	12	
3	12	
4	8	
Total	**40**	

Question 1 (8 points total)

Part a) (4 points)

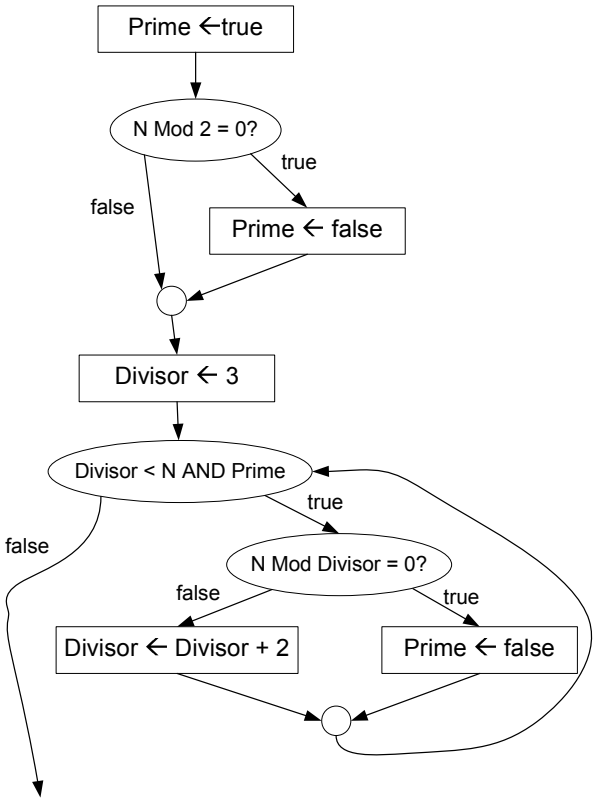


Consider the variables found in memory as shown in the diagram to the right. Note that the variables A and B contain integer values while X and Y contain real values.

For each of the following expressions, write the value produced when evaluated by the computer. Be sure to indicate clearly the type of value (real values should contain a decimal point and Boolean values should be set to true or false).
Examples: `a+1___3__` `x+1 9.2`

Expressions	**Value**
X - (B MOD A)	_____
(T[2] MOD 2 = 0) OR ((A = T[T[4] +1]) AND (NOT (T[1] MOD 2 = 0)))	_____
X + B/2	_____
(T[4]+5.1) = T[1] / y	_____

Part b) (4 points)
Assume N has an initial value of 9, trace the following algorithm (add more rows to the table as needed)



	N	Divisor	Prime
Initial values			

Question 2 (12 marks)

Part a) (6 points, 2 points each)

Circle the SYNTAX and LOGIC errors for the followings:

a) Find three errors in this code:

```
int x;

if(x = 0);
   System.out.println("Your number is 0");
else
{
      if (x > 1)
            System.out.println("Your number is positive");
      else
            System.out.println('Your number is negative');
}
```

b) Find three errors in this code:

```
int choice, num1, num2;

do
{
  System.out.println("Enter a number: ");
  num1 = ITI1120.readInt();
  System.out.println("Enter another number: ");
  num2 = ITI1120.readInt();
  System.out.println("Their sum is " , (num1+ num2));
  System.out.println("Do you want to repeat? Enter 1 if yes, 0 if no");
  choice = ITI1120.readInt();
} while (choice == 0)
```

c) Find three errors in this code:

```
int [] g = new int[]{1,2,3,4,5};
int product;
int i = 0;

while(i <= 5)
  {
      product = product * g[i];

  }

System.out.println ("The product is:" + product);
```

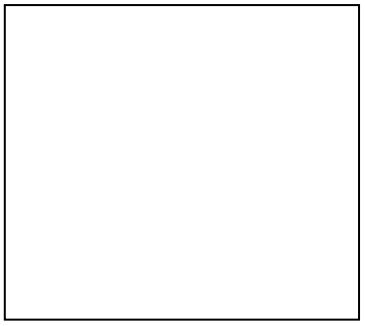
Part b) (6 points, 2 points each)

Find the output of the following Java code:

```
int x = 130;

if(x >= 0 && x <= 100)
   System.out.println ("Choice 1");
else
{
  if (x < 0)
      System.out.println ("Choice 2!");
  else
      System.out.println ("Choice 3!");
}
```

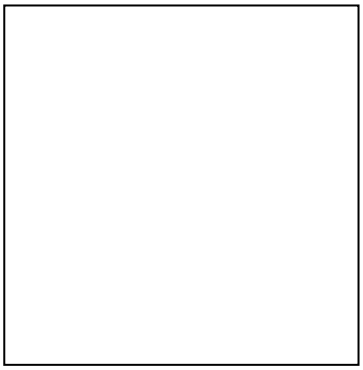
Output Screen



```
  int counter = 1;

  while (counter <= 10)
  {
   counter ++;
   if(counter%3 == 0)
    {
     counter ++;
     System.out.println(counter);
    }
  }
```

Output Screen

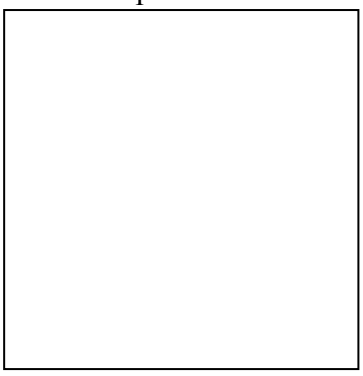


```
  int i, j;

      for(i=3; i>=0 ;i=i-1)
      {
            for(j=0; j<i; j=j+1)
                System.out.print("**");

            System.out.println("");
      }
```

Output Screen



Question 3 (12 marks, 6 points each)

Translate the algorithm found in appendix A on page 10 to a Java program by completing the partially filled methods below and on the next page.

```
import java.util.Scanner; // if you want to use the scanner!

class Q3
{
  public static void main(String [] args)
  {
    // setup the input
    Scanner keyboard = new Scanner(System.in);
    // Declare variables









    // Constraints:  user must enter min of 10
    // Get num from the user

















    // Call the lockerPuzzle method


    // Print the results




  }
```

```
  public static int [] lockerPuzzle(int n)
  {
     // Givens: n - number of lockers/students
     // Results



     // Intermediates









     // Body

```

```


     // return results


  }
}
```

Question 4 (8 marks)
The weight of a child was noted in each of his/her birthdays, starting from age 0 (the day of the birth of the child) until his/her actual `Age`, and these weights are stored in an array called `Weights`.

Develop an algorithm which will determine in what year the percentage of the weight change of the child is the largest (between two birthdays). The algorithm receives the `Weights` array as input and returns the year at which the percentage of weight increase is maximum. For instance, if the largest increase is between 13^{th} and 14^{th} birthdays, the result of algorithm should give 14.

The percentage of increase is calculated in comparison with the previous year weight. For instance, if the weight of the child in her 13^{th} birthday is 40 kg and in its 14^{th} 45 kg birthday, then the percentage of increase is (45-40)/40 = 0.125 or 12.5 %. Make sure you include the attributes of the algorithm (such as Givens, Results, Header, etc.).

This page is for calculations and other work (you can detach this page; it will not be marked)

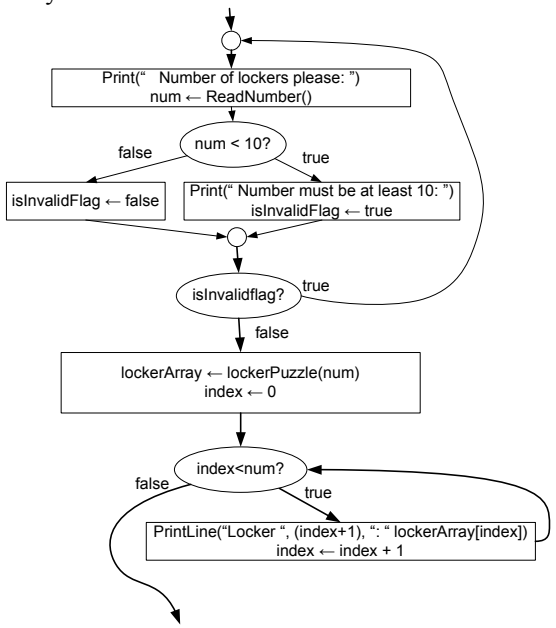
Appendix A: Algorithm for Question 3 (you can detach this page)

Locker Puzzle Software

A school has a number of lockers with an equal number of students (the number is provided by the user). All lockers are closed on the first day of school. As the students enter, the first student, denoted S1, opens every locker. The second student, S2, begins with the second locker, denoted L2, and closes every other locker. Student S3 begins with the third locker (L3) and changes every third locker (closes it if it was open, and opens it if it was closed). Student S4 begins with locker L4 and changes every fourth locker. Student S5 starts with L5 and changes every fifth locker, and so on until student Sn changes Ln (where n is the number of lockers/students given by the user).

The challenge is to find which lockers are open after all students have passed. The following algorithm models provide the logic to get a number from the user (must be at least 10), and to simulate students changing the lockers as they enter the building. The strategy used creates an integer array of "*n*" elements that each represents a locker. An element in the array is incremented when a student changes the corresponding locker. Elements in the array containing odd values at the end of the program execution represent the open lockers.

Intermediates:
num (number of lockers/students)
isInvalidflag (set to true when input by user is invalid)
lockerArray (reference to array of lockers)
index (to index through array to print results)
Constraints: The minimum number of lockers/students is 10
Header: none ← main(none)
Body:



Givens: n (number of lockers/students)
Results: lockersArr (reference to an array)
Intermediates:
lockNum (locker number)
stdntNum (student number)
Header: lockersArr ← lockerPuzzle(n)
Body: