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 2009
Introduction to Computing I
Midterm Examination

Length of Examination: 80 minutes **Feb 25, 2009, 11:30**

Professor: Gilbert Arbez

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: _____

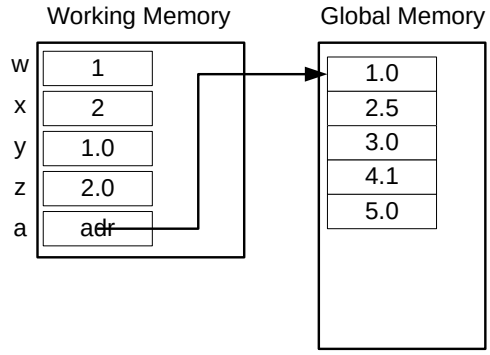
For use of grader:

Question	Marks available	Marks received
1	8	
2	10	
3	12	
4	10	
Total	40	

Question 1 (8 marks total – 4 marks for each part)

Part a)

Consider the variables found in memory as shown in the diagram to the right. Note that the variables w and x contain integer values while y, z and the array 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).

Example: `x+1` 3

Expressions	Value
`w MOD x`	
`a[2] ≥ z AND y ≤ a[0]`	
`w + x/2`	
`(a[4]+5.1) ≥ 6.5 OR a[1] ≠ y`	

Part b)

The following shows the output generated during an execution of the program shown on the next page:

```
Side a
   Please give a value greater than 0: -2.1
   Please give a value greater than 0: 0
   Please give a value greater than 0: 2.5
Side b
   Please give a value greater than 0: 3.8
The hypothenuse is 4.548626
```

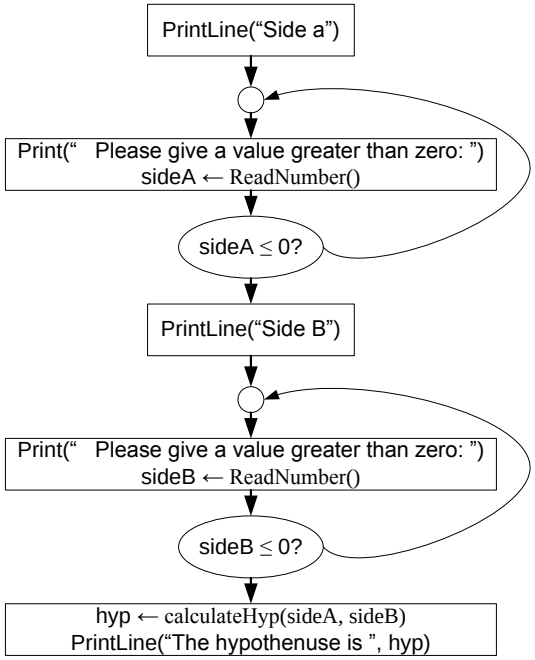
On the following page, show how the contents of the working memory are changed by the execution of the algorithm model. Show how values are passed from one algorithm model to the other using arrows. Record successive assignments to variables/parameters as follows:

Variable ~~2~~, ~~6~~, ~~4~~, 10

Computer Programming Model

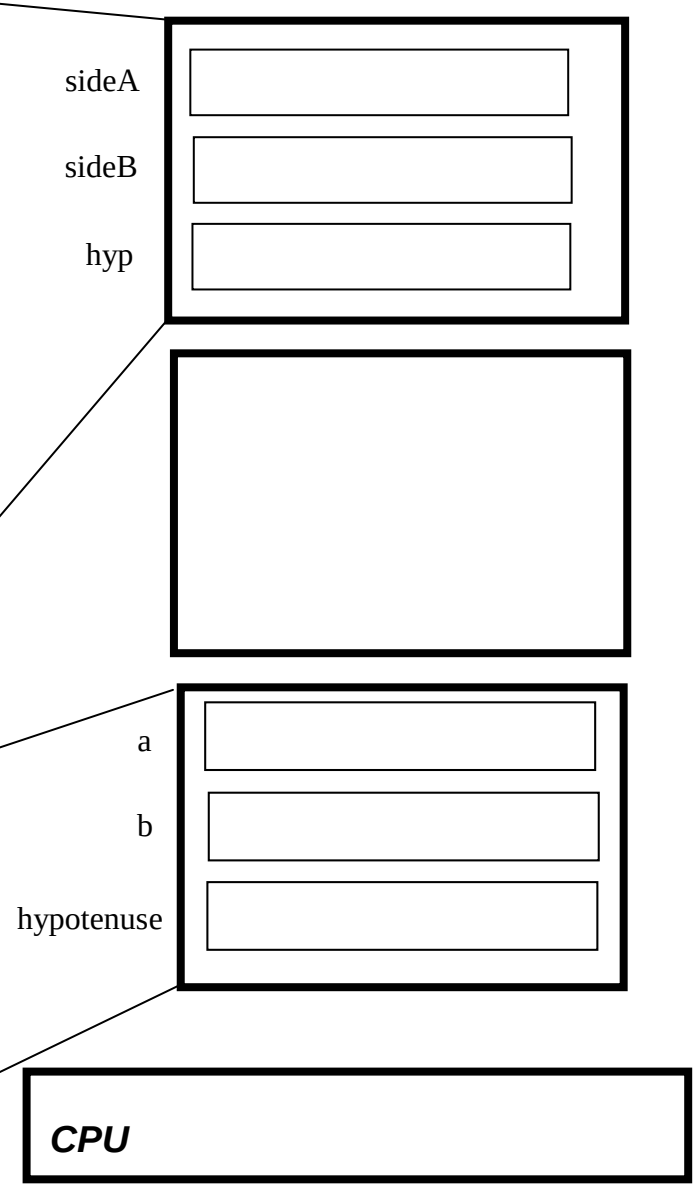
Program Memory

Intermediates:
sideA, sideB (2 sides of a right angle triangle)
hyp (hypothenuse of the triangle
Header: main
Body:



Givens: a, b (side a and side be of a right angle triangle
Results: hypotenuse (length of the hypotenuse
Assumptions: a and b are positive.
Header: hypothenuse ← calculateHyp(a, b)
Body:
hypothenuse ← a*a + b*b
hypothenuse ← $\sqrt{\text{hypotenuse}}$

Working memory



Question 2 (10 marks in total – 4+6 marks)

The following program attempts to reverse the characters inside an array. The main method obtains an array of characters from the user and then calls the method `reverse` to reverse the characters in the array. For example an array containing { ‘a’, ‘b’, ‘c’, ‘d’ } should contain { ‘d’, ‘c’, ‘b’, ‘a’ } after the call to `reverse`.

```
class Q2
{
   public static void main ( String[] args )
   {
      // DECLARE VARIABLES / DATA DICTIONARY
      char[] charArray;    // An array of characters
      int index;           // index into the array

      // Get array from the user
      System.out.print( "Enter an array of characters: " );
      charArray = ITI1120.readCharLine( );

      // Call to reverse the characters entered
      reverse(charArray);

      // PRINT the reversed list
      index = 0;
      while (index < charArray.length)
      {
         System.out.print(charArray [index] + " ");
         index = index + 1;
      }
      System.out.println();
   }
   /* Method: reverse
    * Given: list - an array of characters
    * Description: Reverses the order of characters in the array
    */
   public static void reverse(char [] list)
   {
      // Modified: list
      // Intermediates:
      char [] newList; // new array
      int ix; // index into arrays
      // No results returned
      // Body
      newList = new char[list.length];
      ix=0;
      while(ix < list.length)
      {
          newList[ix] = list[list.length - 1 - ix];
          ix = ix + 1;
      }
      list = newList;
   }
}
```

Part a) The above program does not work, explain why?

Part b) Develop an **algorithm model** for "*reverse*" to correct the logic error presented in part a). DO NOT give a Java method.

Question 3 (12 marks)

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. You do not need to add an identification section to the program.

```
import java.util.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 (10 marks)
The value of e can be approximated using the following series:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + .. + \frac{1}{i!} + ... \frac{1}{n!}$$

Develop an algorithm model that receives *n* (i.e. defines the number of terms used to compute *e*) and returns a value for *e*. Note that the term $1/i!$ is simply $1/i(i\text{-}1)!$, that is the ith term can be computed from the $(i\text{-}1)^{th}$ term simply by dividing it by *i*.

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).

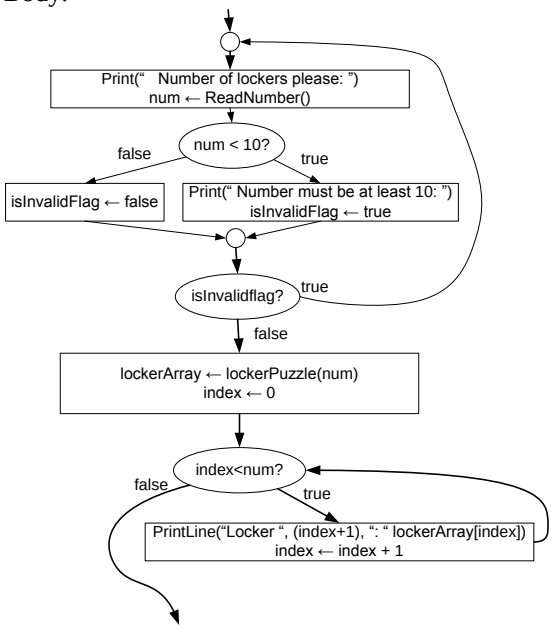
The challenge is to find which lockers are open after all students have passed. The following algorithm models provide the logic to get a number from the user (must be at least 10), and to simulate students changing the lockers as they enter the building. The strategy used creates an integer array of "*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: main

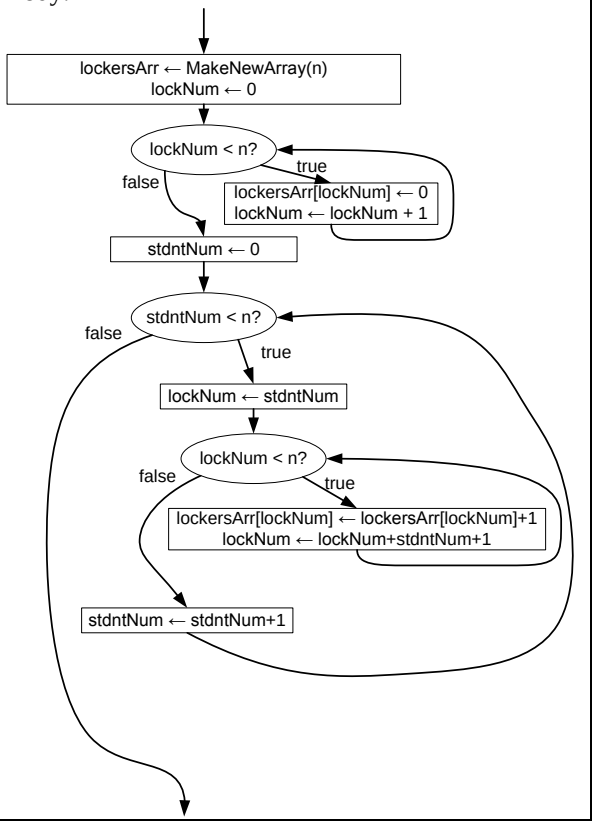
Body:



Givens: n (number of lockers/students)

Results: lockersArr (reference to an array)

Intermediates:
- lockNum (locker number)
- stdntNum (student number)

Header: lockersArr ← lockerPuzzle(n)

Body: