Homework 2 Part 1

CS 1323, Fall 2015

**Name (5 points):**

**Student number:**

This homework is due on Wednesday, September 23 by 11:59 p.m. You must submit homework in a **PDF file** online to the dropbox on Janux. Please make sure that the formatting of the file is appropriate (no empty pages, pages with single words or excessive indentation or large spaces between lines). If you are using Open Office or Star Office, please check to be sure that the figures and diagrams in your PDF file are properly formatted, as this is a common problem with these products.

1. (5 points; 1 point for a), 2 points for b), 2 points for c)) Trace the execution of the following statements by filling in the tables to the right of the statements. The tables may contain too many or too few lines.

|  |
| --- |
| **x** |
|  |
|  |
|  |

* 1. int x = 7;

if (x > 5)

x = 14;

else

x = 21;

* 1. int a = 9;

|  |  |
| --- | --- |
| **a** | **b** |
|  |  |
|  |  |
|  |  |

int b = 5;

if (a==b)

b = a+2;

else if (a < b)

b = b + 2;

else

{

b = a + 1;

a = b + 1;

}

* 1. int size = 3;

|  |  |
| --- | --- |
| **size** | **width** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

int width = 2;

if (size < 5)

{

width = width \* 5;

size = 5;

}

else if (size == 3)

{

width = width \* 4;

size = 5;

}

else if (size == 5)

{

width = width \* 3;

size = 9;

}

else

{

width = width \* 2;

size = 10;

}

1. (16 points; 4 points each) Trace the following while loops by filling out the table at the right. The table may contain too many or too few lines. If the loop is an infinite loop, trace the first three iterations and write “infinite loop” below.

|  |
| --- |
| **count** |
|  |
|  |
|  |
|  |
|  |
|  |

* 1. int count = 4;

while (count <=8)

{

count = count + 1;

}

* 1. int sum = 0;

|  |  |
| --- | --- |
| **sum** | **count** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

int count = 10;

while (count >= 6)

{

sum = sum + count-1;

count = count - 2;

}

* 1. int sum = 10;

|  |  |
| --- | --- |
| **sum** | **count** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

int count = 4;

while (count > 8)

{

count = count + 2;

sum = sum + count;

}

* 1. int count = 0;

|  |  |
| --- | --- |
| **count** | **size** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

int size = 1;

while (size < 14)

{

size = size \* count;

count = count + 1;

}

1. (10 points; 5 points for a), 1 point each for b) to f)) Suppose that we have executed the following statements in a program.

String date= new String (“September”);

String anotherDate = date;

String newDate = new String(“SEPTEMBER”);

String otherDate= new String (“SEPTEMBER 9 2015”);

* 1. Draw a memory diagram of the code above. The heap is on the right. Assume that four characters fit in each address.

Main stack frame Heap

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| date | 100 |  |
| anotherDate | 101 |  |
| newDate | 102 |  |
| otherDate | 103 |  |

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
|  | 1000 |  |
|  | 1001 |  |
|  | 1002 |  |
|  | 1003 |  |
|  | 1004 |  |
|  | 1005 |  |
|  | 1006 |  |
|  | 1007 |  |
|  | 1008 |  |
|  | 1009 |  |
|  | 1010 |  |
|  | 1011 |  |

Give the value (either true or false) of each of the statements

* 1. date == anotherDate
  2. date.equals(anotherDate)
  3. date.equalsIgnoreCase(otherDate)

* 1. newDate.equals(otherDate)
  2. newDate == anotherDate