Midterm 1 Solution

CS 1323, Fall 2015, Section 1

Name (printed legibly):

Student number:

Integrity Pledge

On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signed (type initials for student taking examination online):

Answer all programming questions in Java.

Unless otherwise indicated, each part of a problem is worth the same number of points.

Show your work to receive partial credit.

Pay careful attention to whether the question asks for a code fragment or a complete program. Do not write a whole program when you are asked for only a few lines of code.

Also pay attention to whether you need to get input from a user or not. When you do not need to get input, the problem will say something like “you may assume that the variables’ value was set somewhere else”.

Pay careful attention to distinctions like int versus double and String versus char.

Try to move through short problems quickly to leave you sufficient time to write programs and code fragments.

You will have fifty minutes to take the examination.

You do not need to use import statements on any code.

You may abbreviate System.out.println as S.o.p.

1. (10 points; 2 points each)

What type of data (int, double, String, char, or boolean) would you use to store each of the following things? Do not assume that each type is used exactly once.

* 1. The number of whole years since the last lunar eclipse.

int

* 1. The date of the last lunar eclipse.

String

* 1. The answer to the question, “Did you see the lunar eclipse last night Y/N?”

char or String

* 1. Whether you were able to see the moon's eclipse last night or not.

boolean

* 1. The amount of time that the eclipse was visible, stored in days (lunar eclipses last around 70 minutes, so the number will have a decimal).

double

1. (10 points; 2 points each) Give the value computed for each expression below. Pay careful attention to type, especially char versus String and int versus double.
   1. 48 / 5

9

* 1. 48 % 5

3

* 1. 48 / 5.0

9.60

* 1. "13.2" + "2.7"

“132.22.7”

* 1. 48 != 13

true

1. (20 points; 4 points each part) Find the value assigned by each statement below. Show all intermediate steps to get partial credit. Each part is independent, with the values for any variables starting with the ones given below (***do not use the results of a) in b), for example).*** If the expression is not legal in Java, say so.

Be sure to distinguish double and int values by **giving double values a decimal point**, even if it is a zero.

int width = 10;

double height = 2.5;

* 1. width = width \* 2 + 3;

width = 10 \* 2 + 3

= 20 + 3

= 23

* 1. height = width % 3 \* 14 / 3;

=10 % 3 \* 14 / 3

= 1 \* 14 / 3

= 14 / 3

= 4

= 4.0

* 1. double circumference = 2 \* width + 4 \* height;

= 2 \* 10 + 4 \* 2.5

= 20 + 20.0

= 20.0 + 20.0

= 40.0

* 1. width = width - (int) height;

= 10 – (int) 2.5

= 10 – 2

= 8

* 1. width = width - height;

= 10 – 2.5

= 10.0 – 2.5

= 7.5

Cannot be assigned to width--integer

1. (5 points) Your company sells advertisers a location on websites. Advertisers pay a flat fee of $229.99 for their advertisement. They also pay a fee based on how many times the webpage is displayed to potential customers (called a per-click fee). The per-click fee is $1.99 per 1000 clicks. The fee is paid for each full or partial 1000 clicks. So if there were 2560 clicks, the per-click fee would be $3.97.

double cost;

int clicks; // given a value elsewhere

cost = 229.99 + (clicks + 999) / 1000 \* 1.99;

-or-

cost = 229.99 + (int) Math.ceil(clicks/1000) \* 1.99;

1. (10 points) ***Complete the code fragment*** started below that calculates the cost of a plane ticket. The ticket has a base cost and the airline charges $35 for each additional bag. For $50 extra you can get a premium seat. People in premium seats are allowed to check one bag for free. For example, if a passenger had a base cost of $250.25, purchased a premium seat and checked two bags, the cost would be $335.25.

Scanner input; // Assume this is previously constructed

int bagCount;

double base;

String premium;

double cost;

System.out.println(“What is the base cost of your ticket?”); ***// input base below***

base = input.nextDouble();

System.out.println(“How many bags need to be checked?”); ***// input bagCount below***

bagCount = input.nextInt();

System.out.println(“Do you want to pay for premium service? Yes/No”); ***// input premium below***

premium = input.next();

// Perform cost calculation below using values you read in above. Save the result in cost.

cost = base;

if (premium.equalsIgnoreCase(“Yes”))

{

cost = cost + 50;

if (bagCount > 0)

cost += (bagCount – 1) \* 35;

}

else // no premium service

{

cost = cost + bagCount \* 35;

}

1. (15 points) ***Trace the code fragments*** below in the tables at the right. If there is an infinite loop, trace three iterations and write “infinite loop” in the table.

a)

|  |
| --- |
| **size** |
| 5 |
| 3 |
| 6 |
|  |
|  |
|  |
|  |

int size = 5;

if (size > 10)

size = size + 3;

if (size < 9)

{

size = size – 2;

if (size >= 3)

size = 2 \* size;

}

b)

int size = 29;

int sum = 0;

|  |  |
| --- | --- |
| **size** | **sum** |
| 29 | 0 |
| 24 | 24 |
| 19 | 43 |
| 14 | 57 |
| 9 | 66 |
|  |  |
|  |  |

while (size > 10)

{

size = size - 5;

sum = sum + size;

}

c)

|  |  |
| --- | --- |
| **count** | **accumulator** |
| 20 | 0 |
| 25 | 25 |
| 30 | 55 |
| 35 | 90 |
| Infinite | Loop |
|  |  |
|  |  |
|  |  |

int count = 20;

int accumulator = 0;

while (count > 13)

{

count = count + 5;

accumulator = accumulator + count;

}

1. (30 points) You work for the Pear company that produces the jPhone. Their phones are produced in batches of 1000. Some phones from each batch have to be tested to see if the phones can survive a drop from four feet onto cement. This kind of testing is called destructive testing because once the phone has been dropped, whether it is broken or not, it cannot be sold to customers. Pear tests a few jPhones out of every batch of 1000 jPhones produced. The number of phones tested is determined by the history of testing. The first batch has 7 phones tested. If all phones pass, the next batch will test one less phone than the previous batch. When any phone fails to pass the test, the next batch will test one more phone than the previous batch. Under no circumstance are less than five phones or more than ten phones to be tested. The percent of batches accepted and rejected should be reported. The person running the program knows how many batches will be tested today.

Write a ***complete program*** that keeps track of the results of testing. Your program should report the percent of batches that were approved and rejected. The ***exact interaction*** that your program should produce is below. Numbers in italics came from the user.

Welcome to Pear's Testing System

How many batches are to be tested?

*5*

Test 7 jPhones

Did any of the tests fail? Yes/No

*No*

Test 6 jPhones

Did any of the tests fail? Yes/No

*No*

Test 5 jPhones

Did any of the tests fail? Yes/No

*No*

Test 5 jPhones

Did any of the tests fail? Yes/No

*Yes*

Test 6 jPhones

Did any of the tests fail? Yes/No

*No*

Percent of batches approved: 80.0%

Percent of batches failed: 20.0%

**import** java.util.Scanner;

**public** **class** PearTesting

{

**public** **static** **void** main(String[] args)

{

Scanner input = **new** Scanner(System.***in***);

System.***out***.println("Welcome to Pear's Testing System");

**int** numberTested = 7;

**int** passedBatches = 0;

**int** failedBatches = 0;

**int** count = 0;

System.***out***.println("How many batches are to be tested?");

**int** batches = input.nextInt();

input.nextLine();

// This is a counter controlled loop

**while** (count < batches)

{

// Get input from user

System.***out***.println("Test " + numberTested

+ " jPhones next time");

System.***out***.println("Did any of the tests fail? Yes/No");

String answer = input.nextLine();

// Keep track of number to be tested and passes and fails

**if** (answer.equalsIgnoreCase("Yes"))

{

failedBatches = failedBatches + 1;

numberTested = numberTested + 1;

}

**else**

{

passedBatches = passedBatches + 1;

numberTested = numberTested - 1;

}

// Keep numberTested in range

**if** (numberTested < 5)

numberTested = 5;

**if** (numberTested > 10)

numberTested = 10;

count = count + 1;

}

System.***out***.println("Percent of batches approved: "

+ passedBatches\*100/ (**double**) count + "%");

System.***out***.println("Percent of batches failed: "

+ failedBatches\*100 / (**double**) count + "%");

input.close();

}

}