**Reviewing Test**

CS 1A Midterm Exam

|  |  |
| --- | --- |
| Started: | Oct 31, 2014 7:55 PM |
| Finished: | Oct 31, 2014 8:55 PM - auto |

 1 of 1

**CS 1A Midterm Exam     Total Grade: 16.17   (of possible 20 points)**

Finish this test before the due date,***Friday MIDNIGHT, 10/31  11:59 PM***.  Once you begin, you will have 60 minutes to complete it.  If you do not submit before that time, your incomplete exam will be automatically submitted as is. You are not authorized to take this test in multiple sessions, so do not start it unless you have protected time in which to take the test.

You can look at lectures or texts and even use your compiler, but you may not consult any other individuals or non-course help/ask  sites for help.  Reference sites are fine.

Each question is worth 1 point.

Multiple choice questions with square check-boxes may have more than one correct answer.  Multiple choice questions with round radio-buttons have only one correct answer.

Any code fragments you are asked to analyze are assumed to be contained in a program that has all the necessary variables defined and/or assigned.

None of these questions is a trick.  They pose straightforward questions about programming and language concepts and rules taught in this course.

Your Exam has been accepted. Thank you.

(Any further attempts to resubmit this exam will result in an error.)

**Question 1 of 20     Score: 0.67   (of possible 1 point)**

Assuming **num1**is an **int**, which of the following expressions will always give an accurate, double-precision, answer?

(Check all combinations that give double precision accuracy - there will be more than one that should be checked.)

|  |  |  |
| --- | --- | --- |
| Incorrect | A. | answer = (double) ( (num1 + 1) / 100 ); |
| Correct | B. | answer = (double) ( (num1 + 1.00) / 100 ); |
| Correct | C. | answer = ( ( (double)num1 + 1 ) / 100 ); |
|  | D. | answer = (num1 + 1) / 100 ; |
| Correct | E. | answer = (num1 + 1.) / 100 ; |

 Answer Key: B,C,E

**Feedback**

If the quotient is computed using int arithmetic BEFORE casting to double, the fractional data is lost, so the answer will, in general, be wrong.

If either the numerator or the denominator is a double before the quotient is computed, then the answer will be promoted to double and no information lost.

Finally, in the case where either of the terms in the numerator sum is cast to double, then that sum will be promoted to double and the entire numerator will become double, which means the quotient will use double arithmetic.

**Question 2 of 20     Score: 1   (of possible 1 point)**

An "if" statement can be used without an accompanying "else" clause.

Correct True

 False

 Answer Key: True

**Feedback**

"If" statements appear all the time without "else" clauses.  We have seen this in our lessons and in the text.

**Question 3 of 20     Score: 1   (of possible 1 point)**

Match each literal with the associated data type.

|  | Choices - use a choice only once |
| --- | --- |
| A. | '\t' |
| B. | 1.5F |
| C. | 1.5 |
| D. | 1.5p |
| E. | "200.200" |
|  | Match each of the following to a choice |  |  |
| 1. | char | https://myetudes.org/ambrosia_library/icons/correct.png |  |
| 2. | float | https://myetudes.org/ambrosia_library/icons/correct.png |  |
| 3. | (illegal - no valid type) | https://myetudes.org/ambrosia_library/icons/correct.png |  |
| 4. | double | https://myetudes.org/ambrosia_library/icons/correct.png |  |
| 5. | String | https://myetudes.org/ambrosia_library/icons/correct.png |  |

 Answer Key: 1 - A, 2 - B, 3 - D, 4 - C, 5 - E

**Question 4 of 20     Score: 1   (of possible 1 point)**

Not counting static finals (constants), it is normally preferred to pass information to a method by passing arguments to its formal parameter list rather than by using "static class variables (globals)"  for this purpose.

Correct True

 False

 Answer Key: True

**Feedback**

We avoid using "globals" or class variables to pass changeable information between methods (except for ***finals***, which the problem excludes)  because of the large risk of error and exposure.  In Java, class variables are not intended for passing information between methods.  Rather, they represent the intrinsic, implementation-independent data that defines the class.

**Question 5 of 20     Score: 1   (of possible 1 point)**

If a program is written in a messy and inefficient style, it may still compile and run without any compiler or run-time errors.

Correct True

 False

 Answer Key: True

**Feedback**

Unfortunately, it is true that a poorly written and sloppily designed program can compile and run without errors.  However, if and when that program has to be modified by another programmer, such a perfect "working" program shows its weakness, not to mention the weakness of the programmer.

**Question 6 of 20     Score: 0.5   (of possible 1 point)**

Consider the following loop which is an inefficient (at best) attempt to increment **count**200 times:

for ( k = 0; k < 200; k++);

count++;

Check all true statements (there will be more than one).

*Remember:  all variables are assumed to be declared prior to the loop, as needed.*

|  |  |  |
| --- | --- | --- |
|  | A. | It has a compiler error because of the extra semicolon. |
| Correct | B. | It has a logic error because of the extra semicolon. |
| Missed Correct Answer | C. | It has no compiler errors. |
|  | D. | It has both compiler and run-time errors. |

 Answer Key: B,C

**Review**

 Check to review **before** finishing (will be flagged in Table of Contents)

**Feedback**

The semicolon after the***if-condition*** is not a compiler error, but represents an empty if block.  It will compile, but based on the intent of the loop (increment **count**200 times), this semicolon prevents the code from doing what is intended.

Note that in a real program there would probably more statements in the loop body, since just incrementing **count**would be done more easily using a single **count += 200** statement, but that's not relevant to the question.

**Question 7 of 20     Score: 0   (of possible 1 point)**

Consider the code fragment (assumed to be in a program in which all variables are legally defined):

int num1, num2;

double answer;

// program gets num1 and num2 from user, and values received

// are always non-zero ints between -100 and +100 (code not shown)

...

// compute precise quotient:

answer = (double) ( num1 / num2 );

After the assignment statement the variable **answer**, will hold the *most precise quotient possible*, accurate to several digits to the right of the decimal point  ...

|  |  |  |
| --- | --- | --- |
| Incorrect | A. | ...  *ALWAYS, regardless of what* values **num1**and **num2** hold (as long as they are in the stated range). |
|  | B. | ...  *NEVER, regardless of what* values **num1**and **num2** hold. |
|  | C. | ...  *SOMETIMES, i.e., for some, but not all*, values of **num1**and **num2**. |

 Answer Key: C

**Feedback**

If **num1**= 15 and **num2**= 5, then **answer**will be 3.00000000, accurate to several decimal places.

If **num1**= 15 and **num2**= 4, then **answer**will be 3.00000, inaccurate to the right of the decimal place (should be 3.75000).

So, sometimes accurate, sometimes not.

**Question 8 of 20     Score: 1   (of possible 1 point)**

Below there are two descriptions of programming errors.  Match the description with the kind of error it is (either a compiler error  or run-time/logic error).

|  | | Choices - use a choice only once | | |
| --- | --- | --- | --- | --- |
| A. | | Instead of writing the intended  **System.out.println("The expression to be evaluated is 2 + 2");** ,  the programmer accidentally uses single quotes instead of double quotes**);**. | | |
| B. | | In a program that computes an average of 10 numbers you (the programmer) accidentally divide by 9 instead of 10. | | |
|  | Match each of the following to a choice | |  |  |
| 1. | Compiler Error | | https://myetudes.org/ambrosia_library/icons/correct.png |  |
| 2. | Run-time (logic) Error | | https://myetudes.org/ambrosia_library/icons/correct.png |  |

 Answer Key: 1 - A, 2 - B

**Feedback**

* Dividing the wrong number will not affect the compilation.  It is a run-time error
* Single quotes cannot hold the large string needed by**println()**.  Compiler error.

**Question 9 of 20     Score: 0.5   (of possible 1 point)**

Here is a code fragment which appears in a valid **main()**:

int numGifts;

double costPerGift;

// numGifts gets a value somehow in code that is not shown. then ...

if ( numGifts <= 10 )

costPerGift = 3.0;

else if ( numGifts <= 20 )

costPerGift = 2.75;

else

costPerGift = 2.5;

For which values of **numGifts**will **costPerGift**be**2.75** at the end of the fragment?

(Check all boxes that apply - there will be more than one.)

|  |  |  |
| --- | --- | --- |
|  | A. | numGifts = -1 |
|  | B. | numGifts = 8 |
| Incorrect | C. | numGifts = 10 |
| Correct | D. | numGifts = 11 |
| Correct | E. | numGifts = 20 |
|  | F. | numGifts = 30 |

 Answer Key: D,E

**Question 10 of 20     Score: 1   (of possible 1 point)**

Consider the statement:

if ( (ans == 'Y' && errors < 5) || numTries < 10 ) // note uppercase 'Y'

count++;

Which combinations of values results in **count**being incremented after the statement is complete?

(Select *all*that apply.)

|  |  |  |
| --- | --- | --- |
| Correct | A. | ans = 'Y'  (upper case)    errors = 3    numTries = 10 |
| Correct | B. | ans = 'y'     (lower case)    errors = 4    numTries = 5 |
| Correct | C. | ans = 'Y'  (upper case)    errors = 6    numTries = 5 |
| Correct | D. | ans = 'N'    errors = 7    numTries = 5 |
|  | E. | ans = 'Y'  (upper case)    errors = 100    numTries = 22 |

 Answer Key: A,B,C,D

**Question 11 of 20     Score: 1   (of possible 1 point)**

At the ***end*of the third** (***3rd***) loop pass of the following loop:

for (k = 3, count = 2; k < 10; k++)

count++;

what is the value of count?

|  |  |  |
| --- | --- | --- |
| Correct | A. | 5 |
|  | B. | 3 |
|  | C. | 1 |
|  | D. | 6 |

 Answer Key: A

**Feedback**

This is not about the final value of **count**, but the value only part way through the loop, an important intermediate stage to be able to predict.

**Question 12 of 20     Score: 1   (of possible 1 point)**

Consider the code fragment (assumed to be in a program in which all variables are legally defined):

// deal with ages 10-15, inclusive

if (myAge >= 10 |& myAge <= 15)

age\_category\_two = true;

According to

1. the rules of Java,
2. the use of logical and relational expressions, and
3. the published style rules for this course,

this program has the following number and types of errors:

|  |  |  |
| --- | --- | --- |
|  | A. | No style or compiler errors. |
|  | B. | One style error and no compiler errors. |
|  | C. | Two style errors and no compiler errors. |
|  | D. | One style errors and one compiler error. |
| Correct | E. | Two style errors and one compiler error. |
|  | F. | One style error and two compiler errors. |

 Answer Key: E

**Review**

 Check to review **before** finishing (will be flagged in Table of Contents)

**Feedback**

One compiler error:

In a valid **main()** with variables defined (an assumption by the instructions and the problem), this code has one compiler error:  there is no operator |&.

Two style errors according to our course rules:

1. The***if-block*** is not properly indented.
2. The variable **age\_category\_two** does not use **camelCase**naming.

**Question 13 of 20     Score: 1   (of possible 1 point)**

Consider the code fragment (assumed to be in a program in which all variables are legally defined):

if (numTrips > 3)

{

errors++;

chancesLeft--;

}

Using the techniques taught in this course, determine how many tabs this fragment contains.

|  |  |  |
| --- | --- | --- |
|  | A. | It contains no tabs. |
| Correct | B. | It contains one tab. |
|  | C. | It contains two  tabs. |
|  | D. | It contains three  tabs. |

 Answer Key: B

**Feedback**

By click-dragging, we see the tab in front of the statement **chancesLeft--;**

**Question 14 of 20     Score: 1   (of possible 1 point)**

Consider the statement:

if ( (ans == 'Y' || errors < 5) && numTries < 10 ) // note uppercase 'Y'

count++;

Which combinations of values results in **count**being incremented after the statement is complete?

(Select *all*that apply.)

|  |  |  |
| --- | --- | --- |
|  | A. | ans = 'N'    errors = 3    numTries = 10 |
| Correct | B. | ans = 'y'     (lower case)    errors = -4    numTries = -15 |
|  | C. | ans = 'N'    errors = 5    numTries = 5 |
| Correct | D. | ans = 'Y'  (upper case)    errors = 6    numTries = 5 |
| Correct | E. | ans = 'Y'  (upper case)    errors = 100    numTries = -1 |

 Answer Key: B,D,E

**Question 15 of 20     Score: 1   (of possible 1 point)**

Which of the following meanings of the plus sign, +, are built-in to the language (and/or standard libraries or classes that we have been using with the language)?

|  |  |  |
| --- | --- | --- |
| Correct | A. | Numeric addition |
|  | B. | The logical "or" operation |
| Correct | C. | String concatenation |
|  | D. | Numeric subtraction |

 Answer Key: A,C

**Feedback**

We have seen, in our lessons, that both addition and String concatenation are implemented using the + operator.  Neither subtraction nor logical "or" use the + sign.

**Question 16 of 20     Score: 1   (of possible 1 point)**

In a statement that has multiple "else if" clauses, each "else if" should always be indented to the right of the one above it (i.e., should always use*progressive indentation*.)

 True   
Correct False

 Answer Key: False

**Feedback**

There are many times when all the "else if" clauses should be aligned with each other and not indented.  Examples have been given in the lesson on the if/else statement.

**Question 17 of 20     Score: 1   (of possible 1 point)**

The source code that a programmer writes which describes the exact details of a method's implementation is called the "***method definition***".

Correct True

 False

 Answer Key: True

**Feedback**

The code that contains the details about how a method accomplishes its goal is called the ***method definition***.  The method ***invocation***(or method ***call***) is used by a client or **main()** to get the method to run, but it gives us no information about the exact implementation of the method.

**Question 18 of 20     Score: 1   (of possible 1 point)**

Consider the following method definition:

int update( int userIn )

{

userIn++;

return userIn;

}

Which of the following client calls will always result in the variable **clientVal**being incremented by exactly 1?

|  |  |  |
| --- | --- | --- |
|  | A. | clientVal = update( y ); |
| Correct | B. | clientVal = update( clientVal ); |
|  | C. | y = update( clientVal ); |
|  | D. | update( clientVal ); |

 Answer Key: B

**Feedback**

Since the client argument passed into the method sets the value of  the *local variable,***userIn***,* which is *distinct*from that client variable, the method statement**userIn++** *does not affect*the client argument directly.  It is only the***return value*** that affects the client value, and that happens only if the client variable "catches" the result as a functional return.  Since we are asking that the method increment the client variable **clientVal**, we have to both

1. pass **clientVal**to the method, and
2. capture the return into that same variable, **clientVal**

in order to get a reproducible "increment" out of the method.

**Question 19 of 20     Score: 0.5   (of possible 1 point)**

*Using only the topics THROUGH WEEK FIVE (and allowed in our guidelines)*, this program repeatedly asks the user for an input choice until the user enters a "legal" (i.e., acceptable to the program designer) value.  This program might accomplish this using (check any options that would work):

Assume **no method (function)**calls are used in the program.

|  |  |  |
| --- | --- | --- |
|  | A. | An "if" statement but no loop. |
| Correct | B. | A loop and an "if" statement. |
|  | C. | A long sequence of  "if" statements, with no loop. |
| Missed Correct Answer | D. | A loop but no "if" statement. |

 Answer Key: B,D

**Feedback**

* You must have a loop in order to repeatedly ask the user for input - if the user gives incorrect input for 100 times, then you would have to repeat the question 100 times, but there is no way to know this before the program is run - therefore you would need a while or for loop.  Recursion, methods or other illegal techniques (a hypothetical goto statement) were expressly disallowed in the question.
* In order to test that the user entered a correct answer, you would have to have some logical test.  This could be done either with an ***if***statement or inside the test portion of a ***for***or ***while***statement (obviating the need for an if). It could also be done in a switch statement in the loop.

**Question 20 of 20     Score: 0   (of possible 1 point)**

Here is an expression which contains the + operator in several places on the ***RHS***of an assignment statement (and we don't need to know anything about the shaded variable on the ***LHS***to answer the question):

someVar = **'1' + '2' + '3'** ;

In this expression ...

|  |  |  |
| --- | --- | --- |
|  | A. | ... the + operator represents addition. |
| Incorrect | B. | ... the + operator represents **String**concatenation. |

 Answer Key: A

**Feedback**

The operands are all **chars**, and **chars**are small **ints**, not **Strings**.    The + operator will add the ASCII values, not concatenate the individual digits.

 1 of 1