Tutorial letter 201/2/2017

Introduction to Programming I
COS1511

Semester 2

School of Computing

This tutorial letter contains the discussion of the solutions to assignment 1 semester 2.

DISCUSSION OF ASSIGNMENT 1

Dear Student

Assignment 1 was a multiple choice assignment and was marked electronically. In order to have been able to answer the questions, you should have installed the software (provided on DISK 2017 and downloadable from the Internet) and worked through Lessons 1 to 13 of the Study Guide where you had to write several C++ programs.

Please take the trouble to compare your answers to those given in this tutorial letter. Try to understand why your answer is wrong if it differs from ours. This will help you in acquiring the skill (or art) of programming. The process of becoming a good programmer, involves building new skills and new knowledge and new insights on top of others. This means that it is a growing process and all the building blocks are important.

This assignment is a multiple choice assignment for COS1511. Assignment 2 is a 'written' assignment. In Assignment 2 you have to submit printouts of programs and their output and Assignment 3 is again a multiple choice assignment.

Question 1

Which of the following is a valid variable name?

```
1. 4set
2. four_set
3. 4_set
4. dollar$
```

Correct option: *2*

- Options 1 and 3 are incorrect because a variable name cannot start with a numerical value.
- Option 4 is incorrect because a variable name may not contain an ampersand character.

Question 2

Which of the following is not a valid variable name?

```
1. switch
2. myFloat
3. myDouble
4. sum3
```

Correct option: *1*

- Options 1 is correct as `switch` is a reserved word in C++ and cannot be used as a variable name.
- Options 2, 3 and 4 are all valid variable names.

Question 3

What is the value of x after the following statements?

```
int x, y, z;
y = 10;
z = 3;
x = y * z + 3;
```

1. not defined
2. 60
3. 30
4. 33

Correct option: *4*

Options 4 is correct as * has precedence over +:

```
x  =  y   *  z  +  3
x  =  10  *  3  +  3
x  =     30     +  3
x  =        33
```

You can also include brackets for easier readability:

```
x  =  (y   *  z)  +  3
x  =  (10  *  3)  +  3
x  =     30       +  3
x  =        33
```

Question 4

What is the value of x after the following statements?

```
int x;
x = 0;
x = x * 30;
```

1. 0
2. 30
3. 33
4. not defined

Correct option: *1*

Options 1 is correct as the integer value 0 is assigned to x and thus any multiplication by 0 is 0;.

```
x  =  x   *  30
x  =  0   *  30
x  =     0
```

Question 5

What is the value of `x` after the following statements?

```
int x;
x = x + 30;
```

1. 0
2. 30
3. 33
4. not defined

Correct option: *4*

Options 4 is correct as there is no value assigned to `x`. The program will compile, but the output will be undefined. To test you can write a program as follows:

```
#include <iostream>
using namespace std;

int main()
{
    int x;

    cout << "x + 30 = " << x + 30 << endl;
    return 0;
}
```

When running the program, something similar as below will be displayed.

```
x + 30 = 4309740

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

If `x` was assigned a value, one of the first 3 options could have been a possibility.

Question 6

What is the output of the following code?

```
int value;
value = 33.5;
cout << value << endl;
```

1. 33.5
2. 33
3. value
4. not defined

Correct option: *2*

- Options 2 is correct as `value` is declared as an int value and int values have no decimal point values.
- Option 1 is an float value.
- Option 3 is a string value.

Question 7

What is the output of the following code?

```
float value;
value = 33.5;
cout << "value" << endl;
```

1. 33.5
2. 33
3. value
4. not defined

Correct option: *3*

Options 3 is correct as the `cout` statement outputs the sting `value` as it is in double quotes. The syntax of the `cout` requires opening and closing double quotes " " to display exact text.

`cout << value << endl;` would result in Option 1 being correct.

Option 2 is an integer value.

Question 8

What is the output of the following code?

```
cout << "This is a \\" << endl;
```

1. This is a
2. This is a \
3. nothing, it is a syntax error
4. This is a \ endl

Correct option: *2*

- In order to display a character that has special meaning in C++, e.g. \ or ", an escape character must be used. Thus, as an escape character always start with a \, two \\ must be used.

- To produce Option 1, the `cout` statement would be `cout << "This is a " << endl;`
- To produce Option 2, the `cout` statement would be `cout << "This is a \" << endl;`
- To produce Option 4, the `cout` statement would be `cout << "This is a \ endl";`

Question 9

Which of the following lines correctly reads a value from the keyboard and stores it in the variable named `myInt`?

1. `cin >> myInt;`
2. `cin << myInt;`
3. `cin >> "myInt";`
4. `cin >> myInt >> endl;`

Correct option: *1*

- Option 2, the incorrect operator for `cin` is used: << instead of >>.
- Option 3, the variable `myInt` is in double quotes. The syntax of the `cin` statement does not allow text.
- Option 4, the syntax of the `cin` statement does not allow stream manipulator like `endl`.

Question 10

Which of the following statements is NOT legal?

```
1.  char ch = 'b';
2.  char ch = '0';
3.  char ch = 65;
4.  char ch = "cc";
```

Correct option: *4*

- Option 1 is legal, as a character variable named `ch` is declared and the value `b` is assigned to it.
- Option 2 is legal, as a character variable named `ch` is declared and the value `0` is assigned to it. In this case `0` is a character, not a numeric value.
- Option 3 is legal, as a character variable named `ch` is declared and the asci value `65` is assigned to it. The ASCII value 65 is the character 8.
- Option 4 is illegal, as a character variable named `ch` is declared and then a string value `cc` is assigned to it. No assignment can be between two different types.

Question 11

What is the value of `x` after the following statements?

```
float x;
x = 17/5;
```

```
1.  3.75
2.  3.4
3.  3
4.  60
```

Correct option: *3*

The options is not 100% correct. There are two `/` operators; one between two integers and one between two float values. If both types are present, the float `/`operator will be used.

```
x = 17.0/5;      displays 3.4     => float / int => float
x = 17/5.0;      displays 3.4     => int / float => float
x = 17.0/5.0;    displays 3.4     => float / float => float

x = 17/5;        displays 3 and not 3.0    => int / int => int
```

The reason Option 3 displays `3` even though it is a `float` value is that the computer decides the number of decimal places to use. By adding code to ensure that at least one decimal place will be displayed, the correct output will be displayed and therefore the correct option is 3.

```
float x;
x = 17/5;
cout.setf(ios::fixed);
cout.precision(1);
cout << "x = " << x << endl;
```

Question 12

What is the value of `x` after the following statements?

```
int x;
x = 19/5;
```

1. 15
2. 4
3. 3
4. 3.75

Correct option: *3*

- Option 3 is correct as it is `int / int` which yield an answer of type `int`.
- The calculation in Option 1 and 2 are incorrect.
- Option 4 is incorrect as it is a float value.

Question 13

What is the value of `x` after the following statements?

```
int x;
x = 19 % 5;
```

1. 15
2. 4
3. 3
4. 3.75

Correct option: *2*

Remember that the operator `%` operator can be used to calculate the remainder of one number divided by another and that the remainder is always a number from 0 up to 1 less than the divisor.

- Option 2 is correct as `19 / 5 = 3` remainder `4`
- The calculation in Option 1 and 3 are incorrect.
- Option 4 is incorrect as it is a float value.

Question 14

What is the value of `x` after the following statement?

```
float x;
x = 3.0 / 4.0 + 3 + 2 / 5;
```

1. 5.25
2. 5.75
3. 1.75
4. 3.75

Correct option: *4*

There are two issues to take into consideration.

1. The type of the variable
   a. `x` is declared as a `float`
   b. `2` and `5` is integer values (not 2.0)
   c. `2 / 5 = 0.4`
   d. But as it is integer values the result will be 0

2. The order of the operations (p17 of the study guide). In summary, first * and / and then + and -. But if the operations are the same – from left to right

	3.0	/	4.0	+	3	+	2	/	5
Step 1.		0.75		+	3	+	2	/	5
Step 2.		0.75		+	3	+		0	
Step 3.		0.75		+	3				
Step 4.			3.75						

- In all the other Options the calculations are incorrect.

Question 15

Which C++ statements correctly declare a variable called `age`, and prompt a user for his/her age.

```
1.  int ages = 0;
    cout << "Please enter your age: ;
    cin  >> age;

2.  int age = 0;
    cout << "Please enter your age: ";
    cin  >> ages;

3.  int ages = 0;
    cout >> "Please enter your age: ";
    cin  << age;

4.  int age = 0;
    cout <<  "Please enter your age: ";
    cin  >> age;
```

Correct option: *4*

- Option 1: `cin >> age;` variable name `age` incorrect; must be `ages`.
- Option 2: `cin >> ages;` variable name `ages` incorrect; must be `age`.
- Option 3: the incorrect operators for `cin` and `cout` are used: << and >>.

Question 16

Which C++ code below produced the output below?

```
            A
            BC
            D
```

```
1.    cout << "A" << endl;
      cout << "B";
      cout << "C";
      cout << "D" << endl;

2.    cout << "A" << endl;
      cout << "B" << endl;
      cout << "C" << endl;
      cout << "D";

3.    cout << "A" << endl;
      cout << "B";
      cout << "C" << endl;
      cout << "D" << endl;

4.    cout << "A" << endl;
      cout << "B";
      cout << "C";
      cout << "D";
```

Correct option: *3*

- Option 3 is correct. Take note of the use of `endl`.

Question 17

Which C++ statement is the equivalent of the following statement using a single arithmetic assignment operator?

```
sum = sum + 47;
```

```
1.  sum += 47;
2.  sum == 47;
3.  sum +47;
4.  sum += sum;
```

Correct option: *1*

- Option 2 is the relational operator to find out if two operators are equal.
- In Options 3 the value of `sum` will not change.
- In Option 4 the value of `sum` will be added to the existing value of `sum`.

Question 18

How should the average calculation below be modified to compute the correct average of the two numbers?

```
int num1 = 10;
int num2 = 20;
float avg = 0.0;
avg = num1 + num2 / 2;
```

```
1.    avg = (static_cast<int>(num1) + static_cast<int>(num2)) / 2.0;
2.    avg = (static_cast<float>(num1) + static_cast<float>(num2)) / 2.0;
```

```
3.    avg = (num1) + (num2) / 2.0;
4.    avg = (static_cast(num1) + static_cast(num2)) / 2.0;
```

Correct option: *2*

Here both values must be converted to float values making option 2 syntactically correct.

Question 19

What is the value of `x` after the following statement?

```
float x;
x = 5.0 / 2.0 + (4 + 2)/ 6;
```

```
1. 5.75
2. 5.55
3. 3.75
4. 3.50
```

Correct option: *4*

See discussion at question 14.

Question 20

What is the value of `x` after the following statements?

```
float x;
x = 0;
x += 3.0 * 4.0;
x -= 2.0;
```

```
1. 22.0
2. 12.0
3. 10.0
4. 14.0
```

Correct option: *4*

```
x    =>    0
-------------------------
x    =>    0    +  4.0  *  4.0
x    =>    0    +      16.0
x    =>         16.0
-------------------------
x    =>   16.0  -  2.0
x    =>       14.0
```

See question 11 to set the precision of the number of decimals to display.

Question 21

Given the following code fragment and the input value of `3.0`, what output is generated?

```
float tax;
float total;

cout << "Enter the cost of the item\n";
```

```
cin  >> total;
if ( total >= 3.0)
{
  tax = 0.10;
  cout << total + (total * tax) << endl;
}
else
{
  cout << total << endl;
}
```

1. 3
2. 3.3
3. 4.0
4. 4.4

Correct option: *2*

We hope that you typed in the program and executed it. The value of the variables total is 3.0, thus the if statement evaluates to true and the statement

```
cout << total + (total * tax) << endl;
```

is executed.

```
 total  +  ( total  *  tax  )
  4.0   +  (  3.0   *  0.10 )
  3.0   +         0.3
              3.3
```

Question 22

Given the following code fragment and the input value of `2.0`, what output is generated?

```
float tax;
float total;

cout << "enter the cost of the item\n";
cin  >> total;
if (total >= 3.0)
{
      tax = 0.10;
      cout << total + (total * tax) << endl;
}
else
{
      cout << total << endl;
}
```

1. 2.2
2. 2.0
3. 3.3
4. 3.0

Correct option: *4*

We hope that you typed in the program and executed it. The value of the variable `total` is `3.0`, thus the `if` statement evaluates to `false` and the statement

```
cout << total  << endl;
```

is executed.

Question 23

If `x` has the value of `3`, `y` has the value of `-2`, and `w` is `10`, is the following condition true or false?

```
if( x < 2 && w < y)
```

1. `true && true` → `true`
2. `false && false` → `false`
3. `false && false` → `true`
4. `true && false` → `false`

Correct option: *2*

- `(x < 2) => (3 < 2)` evaluates to `false` as `3` is not less than `2`
- `(w < y) => (10 < -2)` evaluates to `false` as `10` is not less than `-2`
- `(false && false)` evaluates to `false`

Question 24

What is the correct way to write the condition `y < x < z`?

1. `((y < x) && (x < z))`
2. `(y < x < z)`
3. `((y < x) && z)`
4. `((y > x) || (y < z))`

Correct option: *1*

- we want `(x > y)` AND `(x < z)`
- `(x > y)` is the same as `(y < x)`
- the operator that represents AND is `&&`

Question 25

Given the following code fragment, and an input value of `3`, what is the output that is generated?

```
int x;
cout << "Enter a value\n";
cin  >> x;
if(x = 0)
{
  cout << "x is zero\n";
}
else
{
  cout << "x is not zero\n";
}
```

1. x is zero
2. x is not zero
3. unable to determine
4. x is 3

Correct option: *2*

The condition in the `if` statement does not evaluate to `true` as it is the **assignment** operator (=) and not the **equal to** operator (==); the `else` statement will thus be executed.

Question 26

Given the following code fragment, and an input value of `5`, what is the output?

```
int x;
if( x < 3)
{
  cout << "small\n";
}
else
{
  if( x < 4)
  {
       cout << "medium\n";
  }
  else
  {
       if( x < 6)
       {
            cout << "large\n";
       }
       else
       {
            cout << "giant\n";
       }
  }
}
```

1. small
2. medium
3. large
4. giant

Correct option: *3*

- The first `if` evaluates to `false`
- In the `else` part, the first `if` evaluates to `true` and thus `large` it displayed.

Question 27

Given the following code fragment, what is the output?

```
int x = 5;

if( x > 5)
  cout << "x is bigger than 5. ";
  cout <<"That is all. ";
cout << "Goodbye\n";
```

1. x is bigger than 5. That is all
2. x is bigger than 5
3. Goodbye
4. That is all. Goodbye

Correct option: *4*

The `if` evaluates to `false` which is not the case. The first and following statements after the `if` construct is then executed which will result in Option 4 to be correct.

Question 28

Given the following code fragment, what is the final value of `y`?

```
int x, y;
x = -1;
y = 0;

while(x <= 3)
{
  y += 2;
  x += 1;
}
```

1. 2
2. 8
3. 6
4. 10

Correct option: *4*

??		*x*	*y*
		-1	0
x <= 3	true	0	2
x <= 3	true	1	4
x <= 3	true	2	6
x <= 3	true	3	8
x <= 3	true	4	10
x <= 3	false		

Question 29

Given the following code fragment, what is the final value of `y`?

```
int x, y;
x = -1;
y = 0;
while(x < 3)
{
  y += 2;
  x += 1;
}
```

1. 2
2. 10
3. 6
4. 8

Correct option: *4*

??		*x*	*y*
		-1	0
x < 3	true	0	2
x < 3	true	1	4
x < 3	true	2	6
x < 3	true	3	8
x < 3	false		

Question 30

What is the output of the following code fragment?

```
int x = 0;
while( x < 5)
   cout << x << endl;
   x ++;
cout << x << endl;
```

1. 0
2. 5
3. 4
4. infinite loop

Correct option: *4*

Please take note that there is no curly brackets, thus the only statement to execute in the `while`-loop is to print the value of `x`. The actual value of `x` is not changed and therefor the loop will keep on executing. It is important here to take note of the following two issues:

- The condition must evaluate to `true` to exit the `while` loop and in order to evaluate to `true` the value of `x` must change.
- The presence of absence of curly brackets `{ }` to indicate the body of the `while` loop is very important.

Question 31

What is the final value of `x` after the following fragment of code executes?

```
int x=0;
do
{
   x++;
} while(x > 0);
```

1. 8
2. 9
3. 10
4. infinite loop

Correct option: *4*

Note that in a `do… while` loop the loop executes at least once. In this case `x` will always be greater than `1` and will continue to execute 'forever'.

Question 32

Given the following code fragment, which of the following expressions is always true?

```
int x;
cin >> x;
```

```
1. if( x < 3)
2. if( x==1)
3. if( x = 1)
4. if( (x / 3) >1 )
```

Correct option: *3*

- In Option 1, 2 and 4, `x` is compared to a value, therefore it can be `true` or `false`.
- In Option 3, `x` is assigned the value 1, therefore it will always be `true`.

Question 33

What is the output of the following code fragment if the value of `myInt` is `0`?

```
int other = 3, myInt;
if(myInt != 0 && other % myInt !=0)
  cout << "other is odd\n";
else
  cout << "other is even\n";
```

1. `other is even`
2. `other is odd`
3. `0`
4. run-time error, no output

Correct option: *1*

See page 123 of study guide.

Question 34

Which of the following are equivalent to `(!(x < 15 && y >= 3))`?

```
1. (x > 15  && y <= 3)
2. (x >= 15 && y < 3)
3. (x >= 15 || y < 3)
4. (x > 15  || y < 3)
```

Correct option: *3*

Remember:

- A **not AND** (`!&&`) becomes a **OR** (`||`)
- `x` must be less than `15` **NOT** means `x` must be greater equal to `15`
- `y` must be greater than and equal to `3` **NOT** means `y` must be less than `3`

Question 35

Which of the following boolean expressions tests to see if `x` is between `2` and `15` (including `2` and `15`)?

```
1. (x >= 2  && x <= 15)
2. (x <= 15 || x >= 2)
3. (2 <= x  || x <= 15)
4. (2 <= x <= 15)
```

Correct option: *1*

Remember:

- Between means including, thus it must be greater and equal (`>=`) and less and equal (`<=x`)
- Between means AND as it is a row of values `{2,3,4,5,6,7,8,9,10,11,12,13,14,15}`
- Therefore `x` must be greater and equal to `2` AND `x` must be less and equal to `15`

Question 36

What is the output of the following code fragment if `x` is `15`?

```
if(x < 20)
     if(x < 10)
          cout << "less than 10 ";
else
     cout << "large\n";
```

1. `large`
2. `less than 10`
3. `nothing`
4. no output, syntax error

Correct option: *1*

The `if` evaluates to `true`, the first statement to execute is another `if` which evaluates to `false` resulting in the `else` part to be executed. Thus Option 1 is correct.

Question 37

What is the output of the following code fragment?

```
int i = 5;
switch(i)
{
       case 0:   i=15;
                 break;
       case 1:   i=25;
                 break;
       case 2:   i=35;
                 break;
```

```
            case 3:   i=40;
            default:  i=0;
        }
        cout << i <<endl;
```

1. 15
2. 25
3. 0
4. 40

Correct option: *3*

The value of `i` is `5`, and as there is no case to handle the value `5`, the default will be executed resulting in assigning the value `0` to `i` - Option 3.

Question 38

What is wrong with the following switch statement?

```
int ans;
cout <<"Type y for yes on n for no\n";
cin  >> ans;
switch (ans)
{
    case 'y':
    case 'Y': cout << "You said yes\n";
                  break;
    case 'n':
    case 'N': cout << "You said no\n";
                  break;
    default: cout <<"invalid answer\n";
}
```

1. `ans` is a `int`
2. `break;` is illegal syntax
3. nothing
4. there are no break statements on 2 cases

Correct option: *1*

The selector (`ans`) must be of an ordinal data type (e.g. `int`, `char`). In this case `ans` is of type `int` which is correct. However, the cases provide for type `char`. Option 1 states that `ans` is of type `int`, which is incorrect as the code will not execute if `ans` is not of type `char`.

Question 39

Which of the following data types cannot be used in a switch controlling expression?

1. `int`
2. `float`
3. `enum`
4. `char`

Correct option: *2*

The selector must be of an ordinal data type. `float` is not an ordinal type.

Question 40

What is the output of the following code fragment?

```
int x = 0;
  {
       int x = 10;
       cout << x <<",";
  }
  cout << x << endl;
```

1. 10, 10
2. 0, 10
3. 10, 0
4. 0, 0

Correct option: *3*

This fragment of code illustrates scope. x is re-declared between the braces { }. The moment the program exits the code between the braces, the first declaration of x is valid again. Therefore the output will be Option 3.

Question 41

What is the output of the following code fragment?

```
  {
       int x = 13;
       cout << x <<",";
  }
  cout << x << endl;
```

1. 13,13
2. 0,13
3. 13,0
4. nothing, there is a syntax error

Correct option: *4*

As x is not declared outside the braces { }, the compiler will complain that x is not defined in the scope.

Question 42

What is the value of x after the following code executes?

```
int x = 10;
if(x++ > 10)
{
     x =13;
}
```

1. 13
2. 11
3. 10
4. 9

Correct option: *2*

x is initialized to 10. The statement x++ is a shortened way to increase the value of a variable by 1. Take note that the ++ is after the variable name. It is the same as

- `x+=1`
- `x= x + 1`

Because it is `x++`, the evaluation of the condition will first be done, and then `x` will be increased. Therefore, the condition in the `if` statement will first be evaluated - `if (10 > 10)`. This will evaluate to `false` and the first statement after the `if` will be executed. The question asks for the value of `x` which did not change and thus Option 3 is correct.

See discussion below.

Question 43

What is the value of `x` after the following code executes?

```
int x=10;
if( ++x > 10)
{
     x =13;
}
```

1. `9`
2. `10`
3. `11`
4. `13`

Correct option: *4*

Because it is `++x`, `x` will be increased and then the evaluation of the condition will first be done. Therefore, the value of `x` will first be increased, followed by the evaluation of the condition in the `if` statement - `if (11 > 10)`. This will evaluate to `true` and the `if` statement is executed (assigning the value `13` to `x`). The question asks for the value of `x` which makes Option 4 correct.

Question 44

How many times is "`Hi`" printed to the screen

```
for(int i=0; i < 14; i++);
     cout << "Hi\n";
```

1. `1`
2. `13`
3. `14`
4. `15`

Correct option: *1*

A closer look will reveal that there is an "`;`" after the `for` statement. This means that although the `for` loop executed 14 times there was no body to execute and nothing happens. Exiting the loop resulted in the next statement to be executed, which printed `Hi` on the screen once.

Question 45

Given the following code, what is the final value of `i`?

```
int i;
for(i=0; i < 4; i++)
{
     cout << i << endl;
}
```

1. 3
2. 4
3. 5
4. 0

Correct option: *2*

Remember from the earlier discussion on `i++`, first the evaluation, then the value of `i` is increased. Therefore, when the value of `i` equals `4`, the `for` loop will be existed making Option 2 the correct selection.

Question 46

If you want a loop to quit iterating if `x < 10` and `y > 3`, what would be the proper loop condition test?

1. `(x >= 10 || y <= 3)`
2. `(x < 10 && y > 3)`
3. `(x > 10 || y < 3)`
4. `(x >= 10 && y <= 3)`

1.
Correct option: *1*

Remember:

- The loop must quit if one of these condition are met.
- `x` must be less than `10` for the loop to exit. To enter the loop the opposite must be true and therefore `x` must be greater or equal to `10` (`x >= 10`).
- `y` must be greater than `3` for the loop to exit. Therefore to enter the loop the opposite must be true and `y` must then be less and equal to `3` (`y <= 3`).
- An **AND** will exit the loop. Thus to enter the loop it will be an **OR** (`||`).

Question 47

If you need to write a **`while`** loop that will ask the user to enter a number between `2` and `5` inclusive, and will keep asking until the user enters a correct number, what is the loop condition?

1. `(2 <= num <= 5)`
2. `(2 > number || number > 5)`
3. `(2 <= number && number <= 5)`
4. `(2 < number || number > 5)`

Correct option: *4*

This question is quite confusing. One characteristic of a do…while loop is that it will always execute at least once. No matter what the first input value of number is, the statements in the loop body will be executed. Thus for this question to make sense it should have asked for a while loop as input must be validated. This question will not be marked. Following is code that you can enter to illustrate the problem.

```
#include <iostream>
using namespace std;
int main()
{
    int number;
    cout << "Please enter a number between 2 and 5 (inclusive)? ";
    cin >> number;

    while (number < 2 || number > 5)
    {
       cout << "Number is " << number << " is not between 2 and 5
                  (inclusive)." << endl;
       cout << "Please re-enter the number ";
```

```
            cin >> number;
        }

        cout << endl;
        cout << "You have entered a valid value " << number << endl;
        return 0;
    }
```

Question 48

Which loop structure always executes at least once?

1. `for`
2. `while`
3. `do-while`
4. sentinel

Correct option: *3*

The `while` condition can evaluate to `false` before the loop is entered.

Question 49

Which of the following are valid case statements in a `switch`?

```
1. case x < 4:
2. case 1:
3. case 'ab':
4. case 1.5:
```

Correct option: *2*

The selector in Option 1 is a condition (`x < 4`) and therefore not an ordinal type.
The selector in Option 3 is a string incorrectly placed in single quotes and a string is not an ordinal type.
The selector in Option 4 is a float value and therefore not an ordinal type.

Question 50

What is wrong with the following for loop?

```
for(int i=0; i < 10; i--)
{
     cout << "Hello\n";
}
```

1. cannot use a `for-loop` for this
2. `i` is not initialized
3. off-by-one error
4. infinite loop

Correct option: *4*

The condition will never evaluate to `true` as `1` is subtracted from `0` and that will always be less than `10`.

Question 51

Given the code below, which of the following loops will continue to accept values for the variable `age`, and print them out, as long as the value entered is greater than zero?

```
                        int age = 0;
```

1.
```
cout << "Enter an age greater than zero: ";
cin >> age;
while (age < 0)  //begin loop
{
     cout << "The age entered was: " << age << endl << endl;
     cout << "Enter an age greater than zero: ";
     cin >> age;
}
```

2.
```
for (i = 0; i < 10; i++)  //begin loop
{
     cout << "Enter an age greater than zero: ";
     cin >> age;
     cout << "The age entered was: " << age << endl << endl;
}
```

3.
```
do  //begin loop
{
     cout << "The age entered was: " << age << endl << endl;
     cout << "Enter an age greater than zero: ";
     cin >> age;
} while (age > 0);
```

4.
```
cout << "Enter an age greater than zero: ";
cin >> age;
do               //begin loop
{
       cout << "The age entered was: " << age << endl << endl;
       cout << "Enter an age greater than zero: ";
       cin >> age;
} while (age > 0);
```

Correct option: *4*

- The condition of the `while` loop in Option 1 will only be entered if a value less than `0` is entered.
- Option 2 is a `for` loop which will execute a pre-determined number of times.
- In Options 3, as it is a `do…while` loop it will execute at least once and will continue until 0 is entered even if the first value entered was 0.

Question 52

Which of the following C++ *do while* statements will print the numbers 1 to 10 one per line on the screen given the declaration `int count = 0`;?

1.
```
do //begin loop
{
      count = count + 1;
      cout << count << endl;
} while (count <=10);
```

2.
```
do //begin loop
{
      cout << count;
      count = count + 1;
} while (count <=10)
```

3.
```
do //begin loop
{
      count = count + 1;
      cout << count << endl;
} while (count < 10);
```

4.
```
do //begin loop
{
      cout << count << endl;
      count = count + 1;
} while (count < 10);
```

Correct option: *3*

In Options 1 will print all integers from `1` to `11` below one another.
In Options 2 will print `012345678910`.
In Options 4 will print all integers from `0` to `9` below one another.

Question 53

Which of the following C++ `do while` statements will display five asterisks on the same line and advances the cursor to the next line when done? Use the declaration below:

```
int count = 1;
```

1.
```
do   //begin loop
{
       cout << "*";
       count = count + 1;
} while (count <= 5);
```

2.
```
do   //begin loop
{
       cout << "*";
       count = count + 1;
} while (count < 5);
cout << endl;
```

3.
```
do   //begin loop
{
       cout << "*";
       count = count + 1;
} while (count <= 5);
cout << endl;
```

4.
```
do   //begin loop
{
       cout << "*";
       count = count + 1;
} while (count <= 5);
cout << endl;
```

Correct option: *4*

In Options 1 will display `*****` but does not advance to the next line.
In Options 2 and 3 will display `****` and advances to the next line.

Question 54

Which of the following C++ `for` statements display five asterisks on the screen on the same line and advances the cursor to the next line when done?

1.
```
for (int count=1; count <= 5; count = count + 1)
      cout << "*";
      cout << endl;
```

2.
```
for (int count=1; count <= 5; count = count + 1)
{
      cout << "*";
      cout << endl;
}
```

3.
```
for (int count=0; count <= 5; count = count + 1)
      cout << "*";
cout << endl;
```

4.
```
for (int count=1; count < 5; count = count + 1)
      cout << "*";
      cout << endl;
```

Correct option: *1*

In Options 2 will display five asterisks below one another.
In Options 3 will display `******`.
In Options 4 will display `****`.

Question 55

Which of the following C++ code segments below displays 5 asterisks in 3 rows?

1.
```
for (int outer = 1; outer <= 5; outer = outer + 1)
{
      for (int inner = 1; inner <= 3; inner = inner + 1)
           cout << "*" ;
           cout << endl;
}
```

2.
```
for (int outer = 1; outer < 5; outer = outer + 1)
{
      for (int inner = 1; inner <= 3; inner = inner + 1)
           cout << "*" ;
}
```

3.
```
for (int outer = 1; outer <= 5; outer = outer + 1)
{
        for (int inner = 1; inner < 3; inner = inner + 1)
           cout << "*" ;
        cout << endl;
}
```

4.
```
for (int outer = 1; outer <= 3; outer = outer + 1)
{
      for (int inner = 1; inner <= 5; inner = inner + 1)
           cout << "*" ;
        cout << endl;
}
```

Correct option: *4*

In Options 1 will display 3 asterisks in 5 rows.
In Options 2 will display `***************` (15 asterisks).
In Options 3 will display 2 asterisks in 5 rows.

Question 56

Which of the following C++ code segments below displays 2 lines containing 6 asterisks each?

1.
```
for (int outer = 1; outer < 6; outer = outer + 1)
{
      for (int inner = 1; inner <= 2; inner = inner + 1)
           cout << "*";
      cout << endl;
}
```

2.
```
for (int outer = 0; outer <= 2; outer = outer + 1)
{
      for (int inner = 1; inner <= 6; inner = inner + 1)
           cout << "*";
           cout << endl;
}
```

3.
```
for (int outer = 1; outer <= 2; outer = outer + 1)
{
      for (int inner = 1; inner <= 6; inner = inner + 1)
           cout << "*";
      cout << endl;
}
```

4.
```
for (int outer = 1; outer <= 2; outer = outer + 1)
{
      for (int inner = 1; inner < 6; inner = inner + 1)
           cout << "*";
           cout << endl;
}
```

Correct option: *3*

In Options 1 will display 2 asterisks in 5 rows.
In Options 2 will display 6 asterisks in 3 rows.
In Options 4 will display 5 asterisks in 2 rows.

Question 57

Which statement represents the condition of the `if` statement below when it is rewritten without using the `!=` operator?

```
if ( number != 50 )
```

```
1. if ( (number < =50) || (number >= 50) )
2. if ( (number < 50)  || (number > 50) )
3. if ( (number < 50)  || (number >= 50) )
4. if ( (number =< 50) || (number > 50) )
```

Correct option: 2

`!=` means anything but `50`, thus any value less than `50` **OR** any value greater than `50`.
The condition for the `if` statement must evaluate to `true` if `number` is greater than `50` **OR** `number` is less than `50`.

Question 58

Which code segment will display "`yes`" if the value stored in the variable *number* is between 1 and 100, inclusive (including 1 and 100)?

```
1. if ( (number > 1) && (number <= 100) )
      cout << "yes" << endl;

2. if ( (number >= 1) && (number <= 100) )
      cout << "yes" << endl;

3. if ( (number > 1) && (number < 100) )
      cout << "yes" << endl;

4. if ( (number >= 1) && (number < 100) )
      cout << "yes" << endl;
```

Correct option: *2*

Including means greater and equal **AND** less and equal.

Question 59

Given the C++ instructions below, which statements will swap the contents of the variables `num1` and `num2`.

```
            int num1 = 10;
            int num2 = 15;
            int temp = 0;

1. temp = num2;
   num1 = num2;
   num2 = temp;

2. num1 = temp;
   num1 = num2;
   num2 = temp;

3. temp = num1;
   num1 = num2;
   num2 = temp;

4. temp = num1;
   temp = num2;
   num2 = temp;
```

Correct option: *3*

Question 60

Which code segment will display "yes" if the value stored in the variable number is between 1 and 100, exclusive (excluding 1 and 100)?

```
1. if ( (number >= 1) && (number =< 100) )
            cout << "yes" << endl;

2.  if ( (number > 1) && (number =< 100) )
        cout << "yes" << endl;

3.  if ( (number >= 1) && (number < 100) )
       cout << "yes" << endl;

4.  if ( (number > 1) && (number < 100) )
      cout << "yes" << endl;
```

Correct option: *4*

Excluding means greater than **AND** less than.

Question 61

Given the following code:

```
            char letter = ' ';
            cout << "Enter a letter: ";
            cin >> letter;
```

The code below ensures that a lowercase x is changed to an uppercase X. Which statements below represent the same code but without using the toupper function?

```
            if ( toupper(letter) == 'X' )
                cout << "The letter entered was either x or X" << endl;

1. if ( (letter = 'x') || (letter = 'X') )
       cout << "The letter entered was either x or X" << endl;

2.  if ( (letter == 'x') || (letter = 'X') )
       cout << "The letter entered was either x or X" << endl;

3.  if ( (letter == 'x') || (letter == 'X') )
       cout << "The letter entered was either x or X" << endl;

4.  if ( (letter = 'x') || (letter == 'X') )
       cout << "The letter entered was either x or X" << endl;
```

Correct option: *3*

One or either of the conditions in Options 1, 2 and 4 are the assignment operator (=) and not the equality operator (==).

Question 62

Given the following code:

```
int x = 1;
int y = 2;
int z = 3;
```

Which of the following presents the how the evaluation is done?

```
if ( (x == 2) || (y == 2) || (z == 2) )
```

1. false or true or false
   false or false
   false

2. false or true or false
   false or false
   true

3. true or true or false
   true or false
   false

4. false or true or false
   true or false
   true

Correct option: *4*

Remember that

- `true` OR `false` evaluates to `true`
- `false` OR `true` evaluates to `true`
- `true` OR `true` evaluates to `true`
- `false` OR `false` evaluates to `false`

Question 63

Given the following code:

```
int testScore = 0;
cout << "Enter the test score: ";
cin >>  testScore;
```

Which code will display "`yes`" if the test score entered is valid, and "no" if it is not. A valid test score would be in the range from 0 to 100, inclusive (including 0 and 100).

```
1. if ( (testScore >= 0) || (testScore <= 100) )
          cout << "yes" << endl;
   else
          cout << "no" << endl;

2.  if ( (testScore >= 0) && (testScore <= 100) )
          cout << "yes" << endl;
   else
          cout << "no" << endl;

3.  if ( (testScore > 0) && (testScore < 100) )
          cout << "yes" << endl;
   else
          cout << "no" << endl;

4.  if ( (testScore > 0) || (testScore < 100) )
          cout << "yes" << endl;
   else
          cout << "no" << endl;
```

Correct option: *2*

Including excludes Options 3 and 4. Option 1 will allow a `testScore` of anything greater than `0` **OR** less than `100` to be accepted; a `testScore` of `-25` or `350` will be accepted.

Question 64

Given the following code:

```
int number = 50;
```

Which code will display "`yes`" and add one to the `number` variable if the value stored in the `number` variable is greater than or equal to 50?

```
1. if ( number > 50)
   {
          cout << "yes" << endl;
          number = number + 1;
   }

2. if ( number >= 50)
          cout << "yes" << endl;
          number = number + 1;

3. if ( number >= 50)
   {
          cout << "yes" << endl;
          number = number + 1;
   }

4. if ( number <= 50)
   {
         cout << "yes" << endl;
         number = number + 1;
   }
```

Correct option: 3

- *Greater or equal to 50* excludes Options 1.
- Option 2 has no `{}` thus will always add 1 to `number` irrespective the value of `number`.
- Option 4 will only enter if value of variable `number` is less or equal to 50.

Question 65

Given the following code:

```
int number = 49;
```

What will the output be after the following statements are executed?

```
if ( number >= 50)
    cout << "yes " << endl;
    cout << number + 1;
```

1. `yes 50`
2. `49`
3. `50`
4. `yes 49`

Correct option: *3*

The value of `number` is `49`. The condition evaluates to `false` and the first statement after the `if` statement is executed which display the value of `number + 1`.