CSE111: Data Structures

Assignment 1

Name: mahros mohamed mahros alqabasy

ID: 223106831

Date: 20-10-2024

1. Introduction to classes and objects

Fill in the blanks in each of the following:

a) Every class definition contains the keyword class followed immediately by the class's name.

b) A class definition is typically stored in a file with the .cpp filename extension.

c) Each parameter in a function header specifies both a(n) type and a(n) parameter name.

d) When each object of a class maintains its own version of an attribute, the variable that represents the attribute is also known as a(n) private data

e) Keyword `public` is a(n) access modifier

f) Return type void indicates that a function will perform a task but will not return any information when it completes its task.

g) Function getline from the `<string>` library reads characters until a newline character is encountered, then copies those characters into the specified `string`.

h) When a member function is defined outside the class definition, the function header must include the class name and the ::, followed by the function name to "tie" the member function to the class definition. scope resolution operator

State whether each of the following is *true* or *false*. If *false*, explain why.

✗ a) By convention, function names begin with a capital letter and all subsequent words in the name begin with a capital letter. in cammelCase, this is a standard.

✓ b) Empty parentheses following a function name in a function prototype indicate that the function does not require any parameters to perform its task.

✓ c) Data members or member functions declared with access specifier `private` are accessible to member functions of the class in which they're declared.

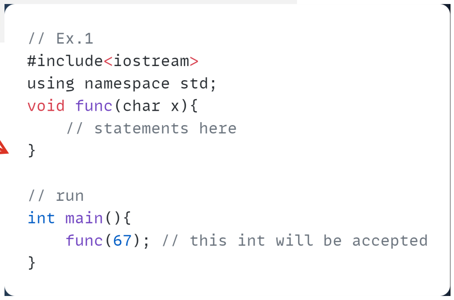
✗ d) Variables declared in the body of a particular member function are known as data members and can be used in all member functions of the class. this is called local variable.

✗ e) Every function's body is delimited by left and right braces (`{` and `}`). except single statement

✓ f) Any source-code file that contains `int main()` can be used to execute a program.

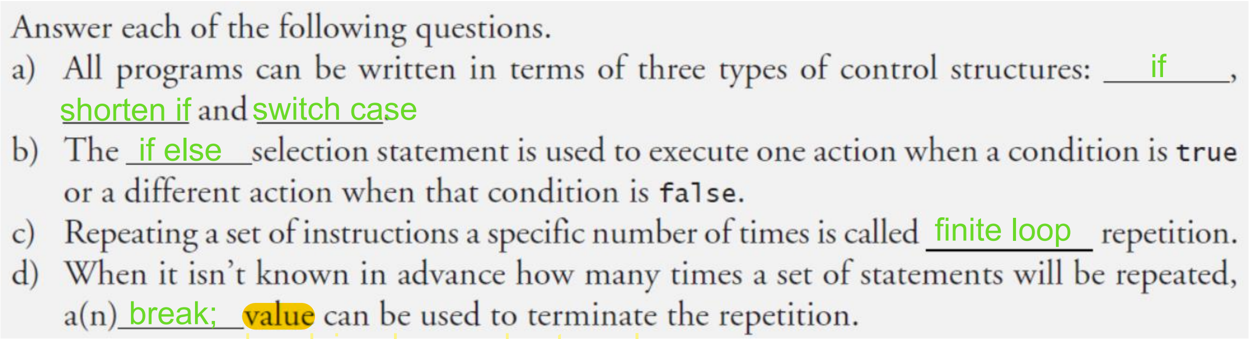
✓ g) The types of arguments in a function call must be consistent with the types of the corresponding parameters in the function prototype's parameter list.

but compiler can make type casting in some cases.



```
// Ex.1
#include<iostream>
using namespace std;
void func(char x){
    // statements here
}

// run
int main(){
    func(67); // this int will be accepted
}
```

2. Control statements: part 1



Answer each of the following questions.

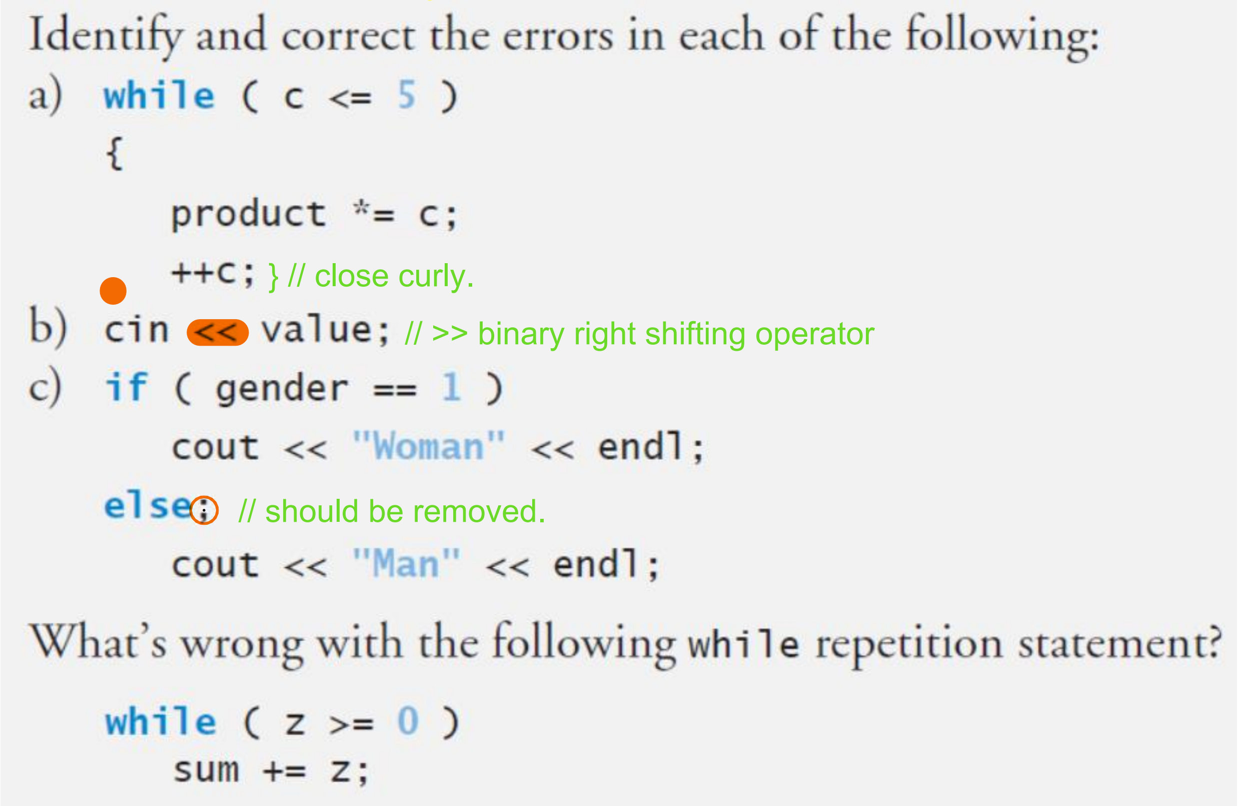
a) All programs can be written in terms of three types of control structures: ___if___, shorten if and switch case

b) The _if else_ selection statement is used to execute one action when a condition is `true` or a different action when that condition is `false`.

c) Repeating a set of instructions a specific number of times is called _finite loop_ repetition.

d) When it isn't known in advance how many times a set of statements will be repeated, a(n) _break;_ value can be used to terminate the repetition.

break is a keyword not a value.



Identify and correct the errors in each of the following:

a)
```
while ( c <= 5 )
{
   product *= c;
   ++c; } // close curly.
```

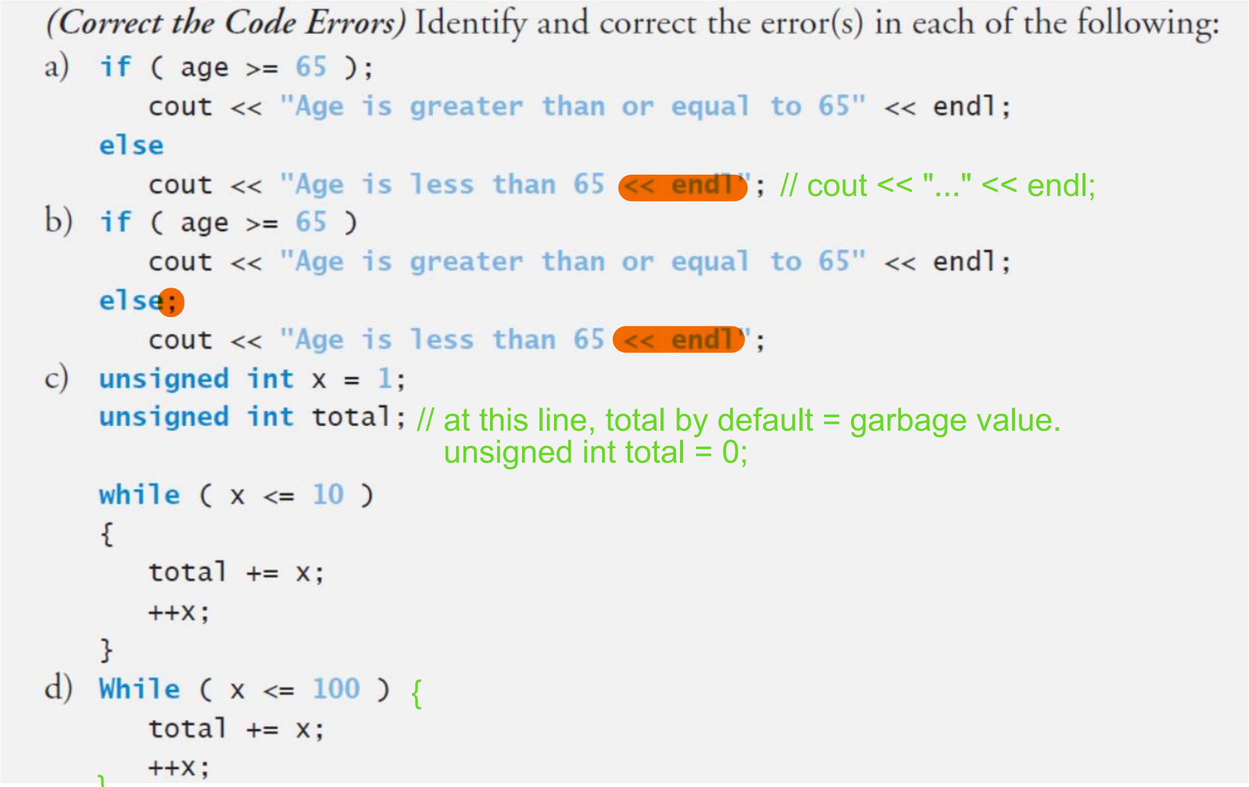
b)
```
cin << value; // >> binary right shifting operator
```

c)
```
if ( gender == 1 )
   cout << "Woman" << endl;
else;  // should be removed.
   cout << "Man" << endl;
```

What's wrong with the following `while` repetition statement?

```
while ( z >= 0 )
   sum += z;
```

there is no any change in the value of z; so this will produce infinite loop; this is logical error.



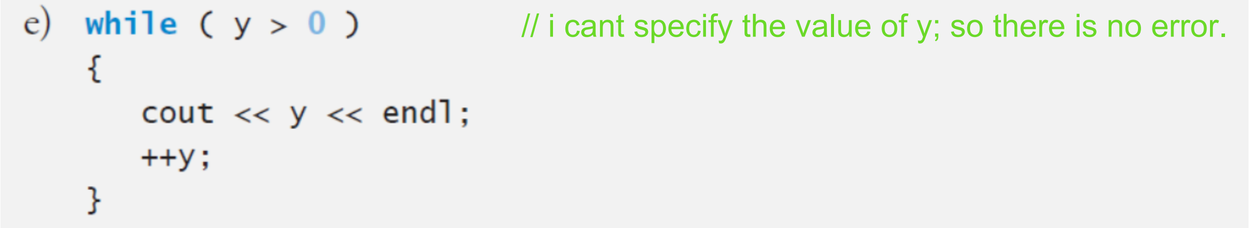
(Correct the Code Errors) Identify and correct the error(s) in each of the following:

a)
```
if ( age >= 65 );
   cout << "Age is greater than or equal to 65" << endl;
else
   cout << "Age is less than 65 << endl"; // cout << "..." << endl;
```

b)
```
if ( age >= 65 )
   cout << "Age is greater than or equal to 65" << endl;
else;
   cout << "Age is less than 65 << endl";
```

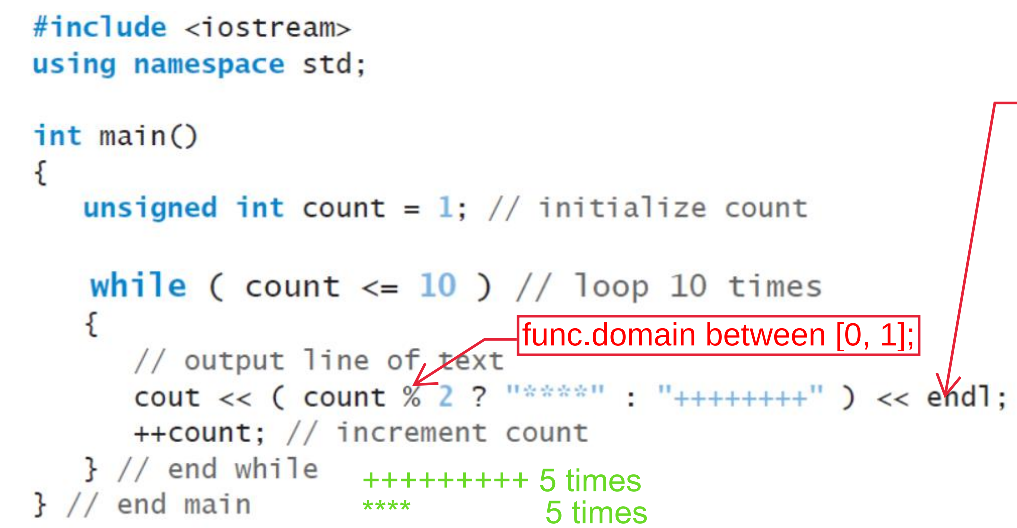
c)
```
unsigned int x = 1;
unsigned int total; // at this line, total by default = garbage value.
                      unsigned int total = 0;

while ( x <= 10 )
{
   total += x;
   ++x;
}
```

d)
```
While ( x <= 100 ) {
   total += x;
   ++x;
}
```



e)
```
while ( y > 0 )               // i cant specify the value of y; so there is no error.
{
   cout << y << endl;
   ++y;
}
```

(What Does this Program Do?) What does the following program print?



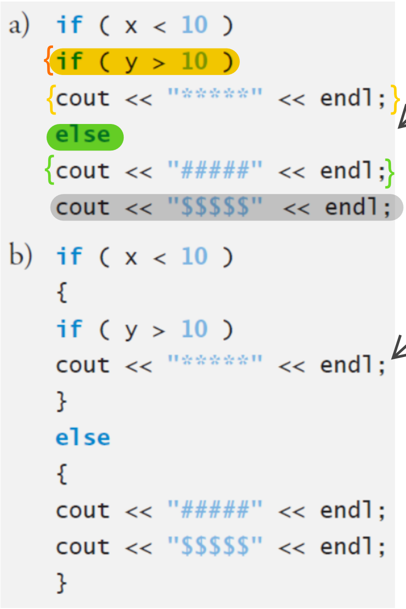
```
#include <iostream>
using namespace std;

int main()
{
   unsigned int count = 1; // initialize count

    while ( count <= 10 ) // loop 10 times
    {
       // output line of text
       cout << ( count % 2 ? "****" : "++++++++" ) << endl;
       ++count; // increment count
    } // end while
} // end main
```

func.domain between [0, 1];

+++++++++ 5 times
**** 5 times

c,	c %2,	output
1	1	****
2	0	++++++++
3	1	****
4	0	++++++++
5	1	****
6	0	++++++++
7	1	****
8	0	++++++++
9	1	****
10	0	++++++++

State the output for each of the following when x is 9 and y is 11 and when x is 11 and y is 9.



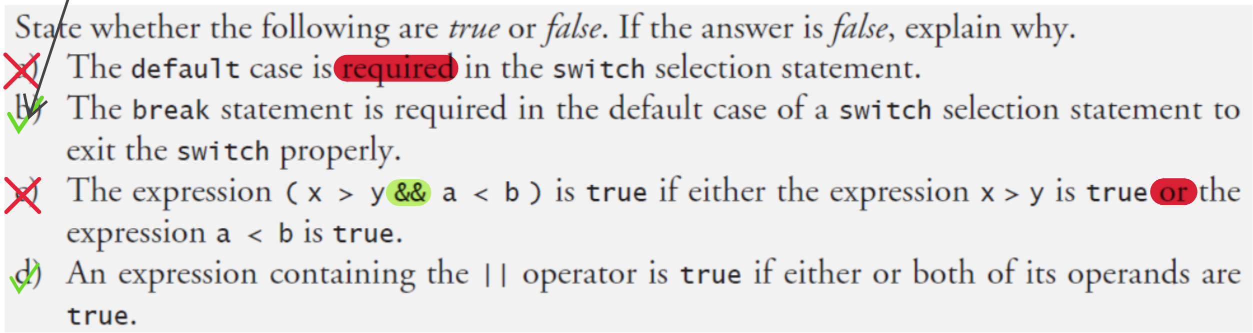
```
a)  if ( x < 10 )
   {if ( y > 10 )
   {cout << "*****" << endl;}}
    else
   {cout << "#####" << endl;}
    cout << "$$$$$"  << endl;

b)  if ( x < 10 )
    {
    if ( y > 10 )
    cout << "*****" << endl;
    }
    else
    {
    cout << "#####" << endl;
    cout << "$$$$$" << endl;
    }
```

```
if(x < 10){ // true
  if(y > 10) cout << "*****" << endl; // *****
}else cout << "#####" << endl; // not happen
cout << "$$$$$" << endl; // $$$$$
```

*****

but in case, we can replace it with return keyword if i am working inside a function. this will not check other cases as break.

3. Control statements: part 2



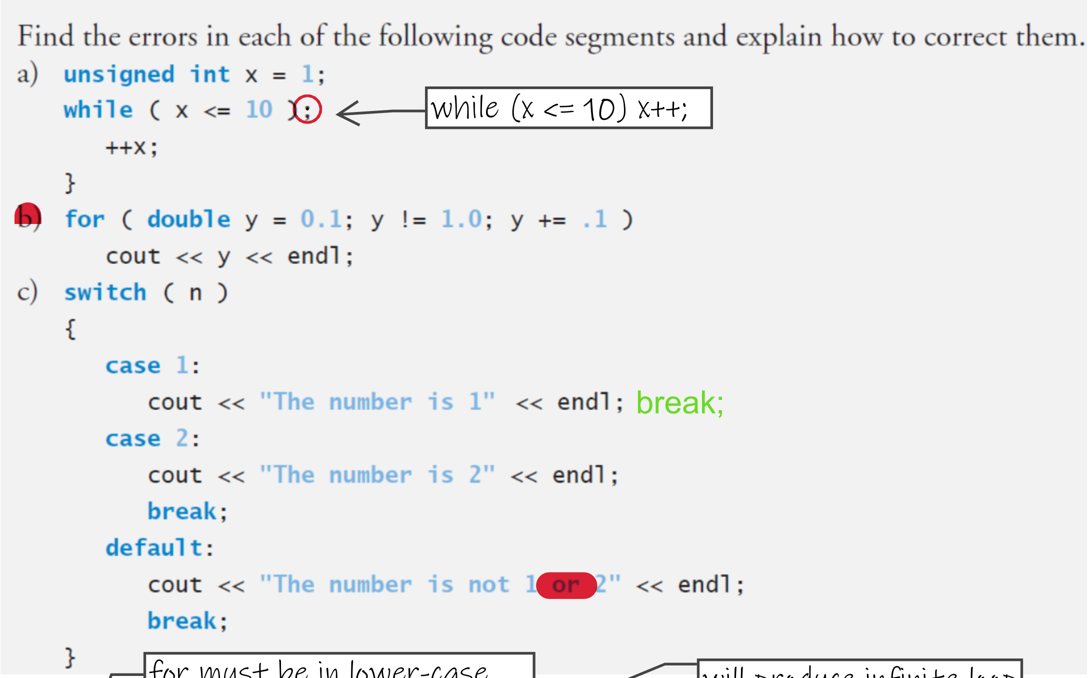
State whether the following are *true* or *false*. If the answer is *false*, explain why.

✗ a) The `default` case is required in the `switch` selection statement.

✓ b) The `break` statement is required in the default case of a `switch` selection statement to exit the `switch` properly.

✗ c) The expression `( x > y && a < b )` is `true` if either the expression `x > y` is `true` or the expression `a < b` is `true`.

✓ d) An expression containing the `||` operator is `true` if either or both of its operands are `true`.



Find the errors in each of the following code segments and explain how to correct them.

a)
```
unsigned int x = 1;
while ( x <= 10 );
   ++x;
}
```

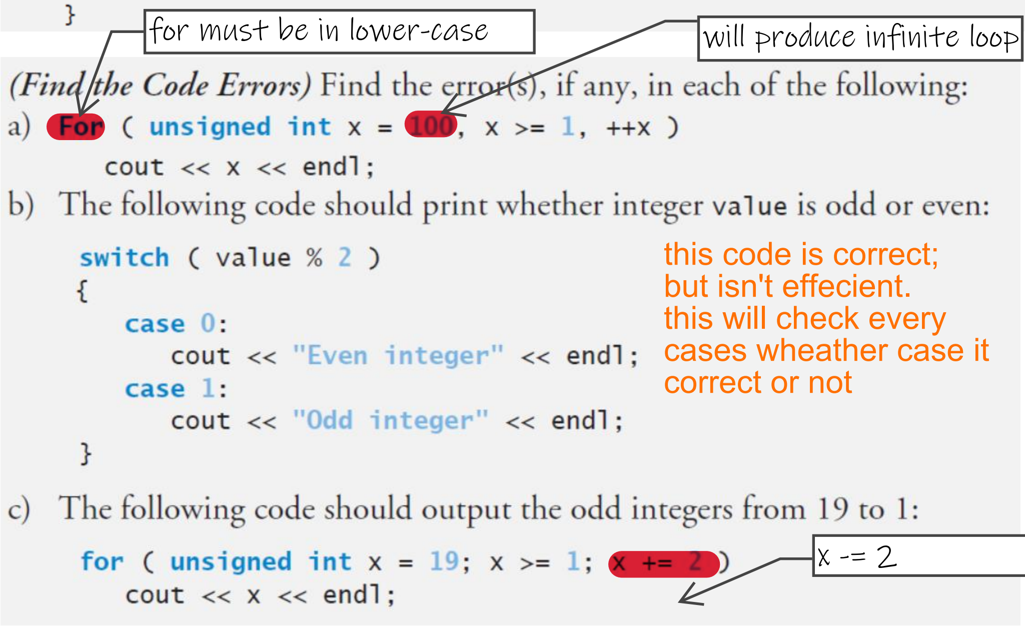
while (x <= 10) x++;

b)
```
for ( double y = 0.1; y != 1.0; y += .1 )
   cout << y << endl;
```

c)
```
switch ( n )
{
   case 1:
      cout << "The number is 1" << endl; break;
   case 2:
      cout << "The number is 2" << endl;
      break;
   default:
      cout << "The number is not 1 or 2" << endl;
      break;
}
```

for must be in lower-case

will produce infinite loop



(Find the Code Errors) Find the error(s), if any, in each of the following:

a)
```
For ( unsigned int x = 100, x >= 1, ++x )
   cout << x << endl;
```

b) The following code should print whether integer `value` is odd or even:

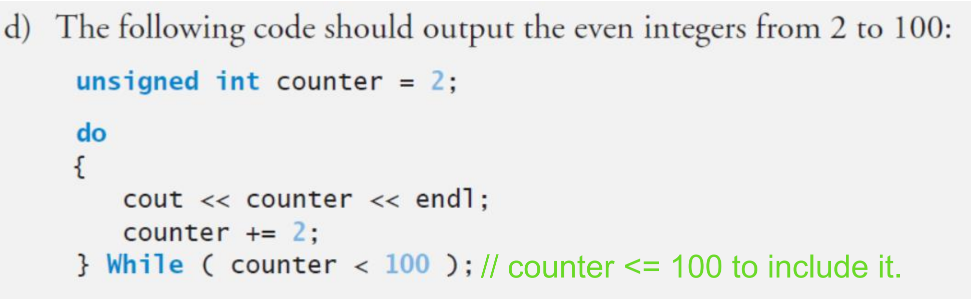
```
switch ( value % 2 )
{
   case 0:
      cout << "Even integer" << endl;
   case 1:
      cout << "Odd integer" << endl;
}
```

this code is correct; but isn't effecient. this will check every cases wheather case it correct or not

c) The following code should output the odd integers from 19 to 1:

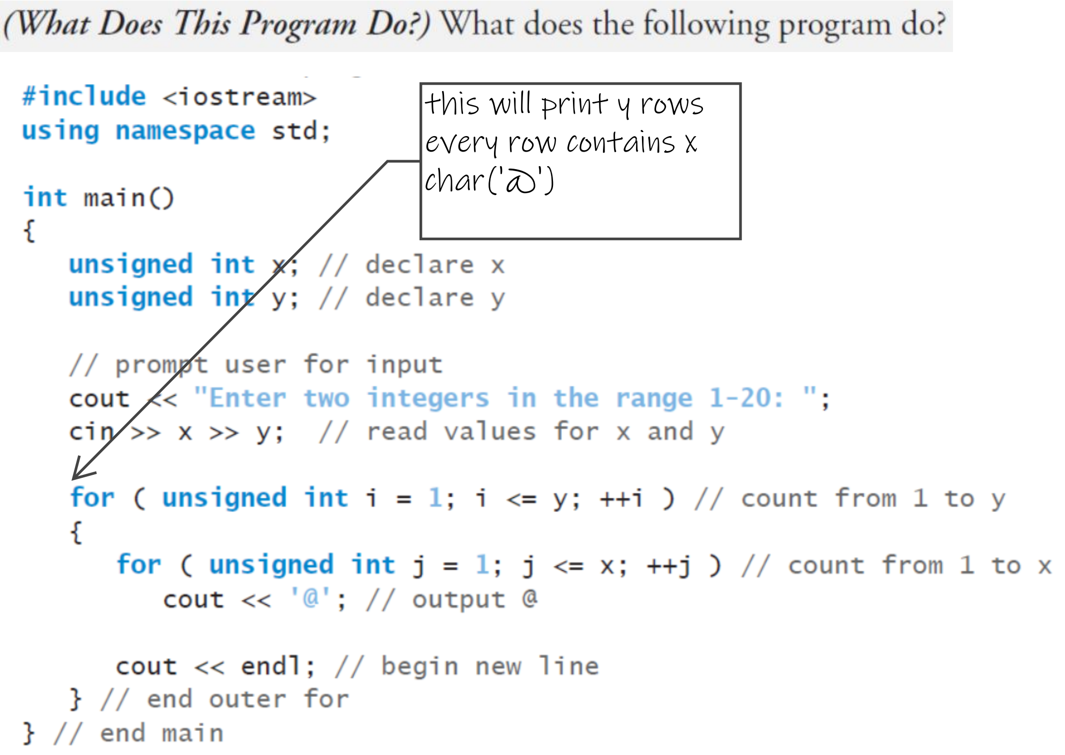
```
for ( unsigned int x = 19; x >= 1; x += 2 )
   cout << x << endl;
```

x -= 2



d) The following code should output the even integers from 2 to 100:

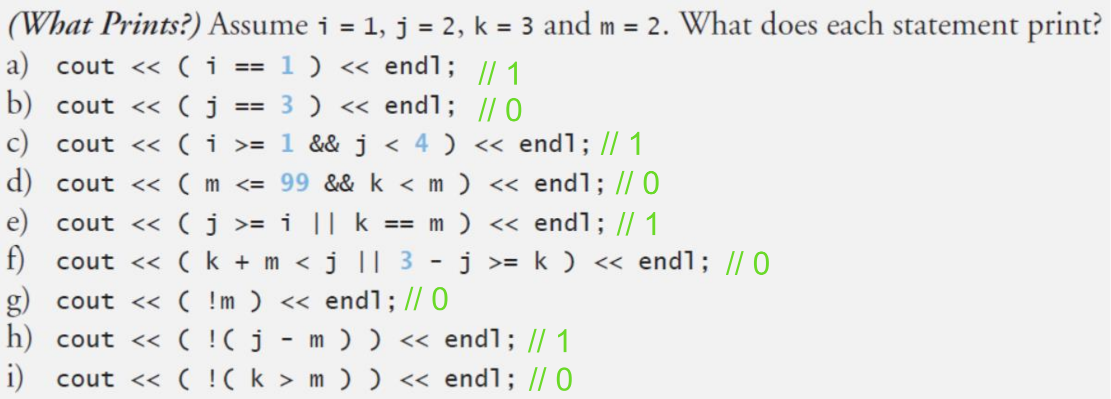
```
unsigned int counter = 2;

do
{
   cout << counter << endl;
   counter += 2;
} While ( counter < 100 ); // counter <= 100 to include it.
```



(What Does This Program Do?) What does the following program do?

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

int main()
{
   unsigned int x; // declare x
   unsigned int y; // declare y

   // prompt user for input
   cout << "Enter two integers in the range 1-20: ";
   cin >> x >> y;  // read values for x and y

   for ( unsigned int i = 1; i <= y; ++i ) // count from 1 to y
   {
      for ( unsigned int j = 1; j <= x; ++j ) // count from 1 to x
         cout << '@'; // output @

      cout << endl; // begin new line
   } // end outer for
} // end main
```

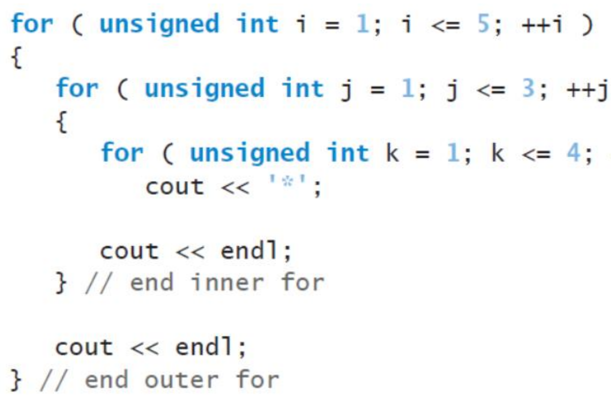
this will print y rows
every row contains x
char('@')



(What Prints?) Assume i = 1, j = 2, k = 3 and m = 2. What does each statement print?

```
a) cout << ( i == 1 ) << endl; // 1
b) cout << ( j == 3 ) << endl; // 0
c) cout << ( i >= 1 && j < 4 ) << endl; // 1
d) cout << ( m <= 99 && k < m ) << endl; // 0
e) cout << ( j >= i || k == m ) << endl; // 1
f) cout << ( k + m < j || 3 - j >= k ) << endl; // 0
g) cout << ( !m ) << endl; // 0
h) cout << ( !( j - m ) ) << endl; // 1
i) cout << ( !( k > m ) ) << endl; // 0
```

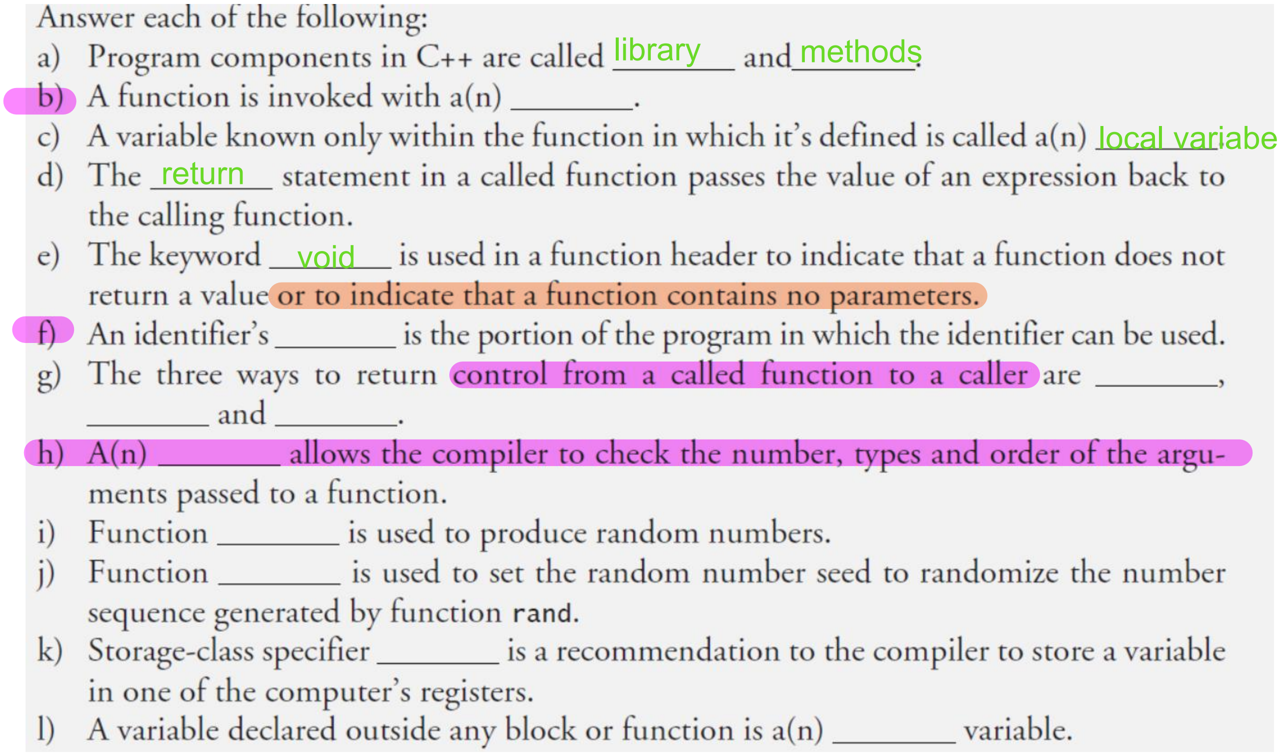
(What Does This Code Do?) What does the following program segment do?



```
for ( unsigned int i = 1; i <= 5; ++i )        // 5 times
{
   for ( unsigned int j = 1; j <= 3; ++j )      // 3 times
   {
      for ( unsigned int k = 1; k <= 4; ++k )     // 4 times
         cout << '*';

      cout << endl;
   } // end inner for

   cout << endl;
} // end outer for
```

```
****
****
****
****
****
****
****
****
****
****
****
****
****
****
```

4. Functions



Answer each of the following:

a) Program components in C++ are called library and methods.

b) A function is invoked with a(n) ________.

c) A variable known only within the function in which it's defined is called a(n) local variabel

d) The return statement in a called function passes the value of an expression back to the calling function.

e) The keyword void is used in a function header to indicate that a function does not return a value or to indicate that a function contains no parameters.

f) An identifier's ________ is the portion of the program in which the identifier can be used.

g) The three ways to return control from a called function to a caller are ________, ________ and ________.

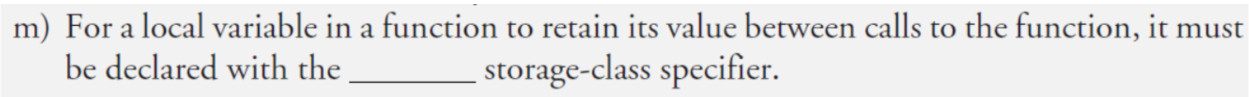
h) A(n) ________ allows the compiler to check the number, types and order of the arguments passed to a function.

i) Function ________ is used to produce random numbers.

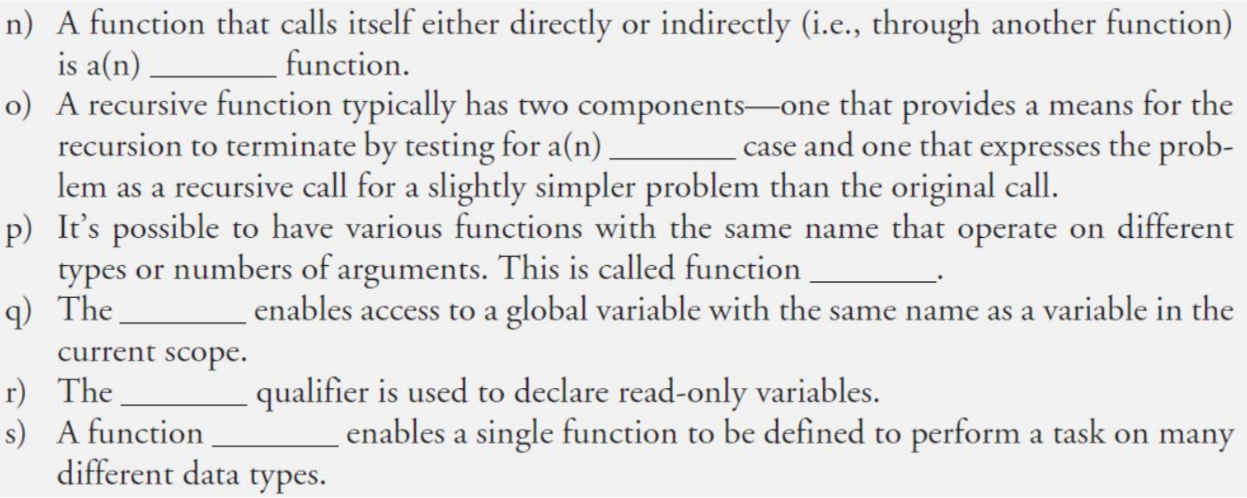
j) Function ________ is used to set the random number seed to randomize the number sequence generated by function `rand`.

k) Storage-class specifier ________ is a recommendation to the compiler to store a variable in one of the computer's registers.

l) A variable declared outside any block or function is a(n) ________ variable.



m) For a local variable in a function to retain its value between calls to the function, it must be declared with the ________ storage-class specifier.



n) A function that calls itself either directly or indirectly (i.e., through another function) is a(n) ________ function.

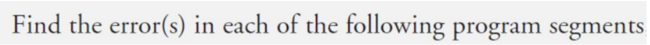
o) A recursive function typically has two components—one that provides a means for the recursion to terminate by testing for a(n) ________ case and one that expresses the problem as a recursive call for a slightly simpler problem than the original call.

p) It's possible to have various functions with the same name that operate on different types or numbers of arguments. This is called function ________.

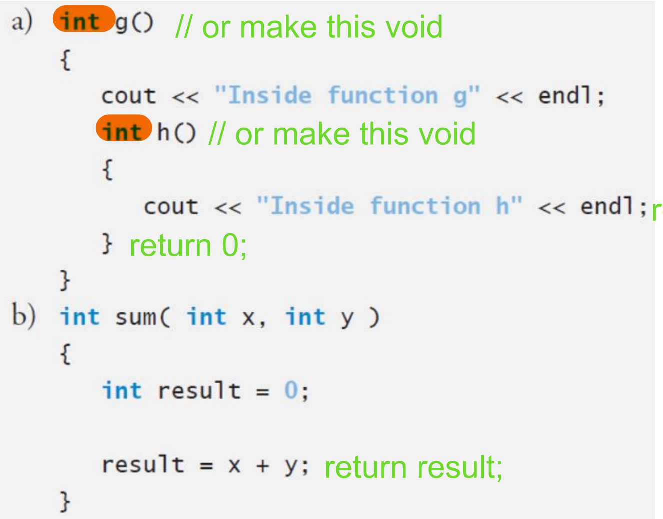
q) The ________ enables access to a global variable with the same name as a variable in the current scope.

r) The ________ qualifier is used to declare read-only variables.

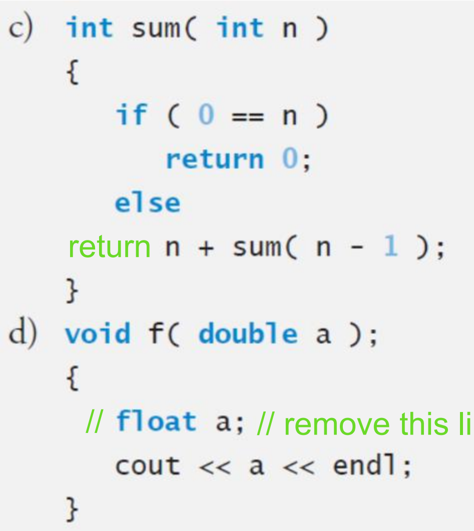
s) A function ________ enables a single function to be defined to perform a task on many different data types.



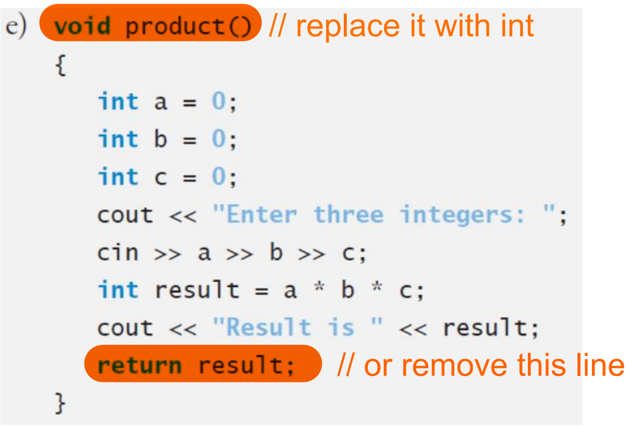
Find the error(s) in each of the following program segments



```
a) int g() // or make this void
   {
      cout << "Inside function g" << endl;
      int h() // or make this void
      {
         cout << "Inside function h" << endl;return 0;
      } return 0;
   }
b) int sum( int x, int y )
   {
      int result = 0;

      result = x + y; return result;
   }
```

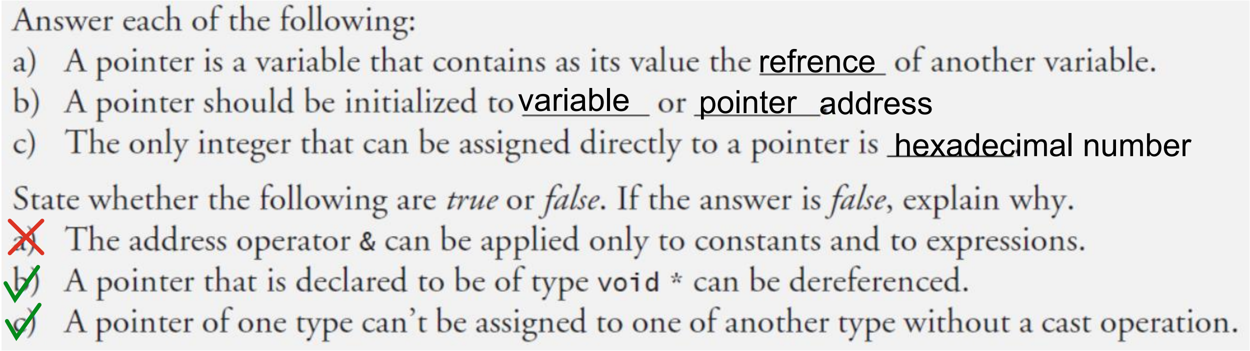


```
c) int sum( int n )
   {
      if ( 0 == n )
         return 0;
      else
   return n + sum( n - 1 );
   }
d) void f( double a );
   {
     // float a; // remove this line
      cout << a << endl;
   }
```



```
e) void product() // replace it with int
   {
      int a = 0;
      int b = 0;
      int c = 0;
      cout << "Enter three integers: ";
      cin >> a >> b >> c;
      int result = a * b * c;
      cout << "Result is " << result;
      return result;  // or remove this line
   }
```

5. Pointers



Answer each of the following:

a) A pointer is a variable that contains as its value the refrence of another variable.

b) A pointer should be initialized to variable or pointer address

c) The only integer that can be assigned directly to a pointer is hexadecimal number

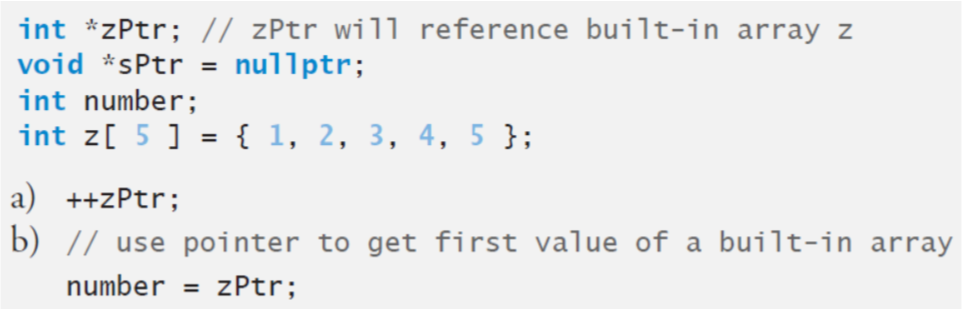
State whether the following are *true* or *false*. If the answer is *false*, explain why.

✗ a) The address operator & can be applied only to constants and to expressions.

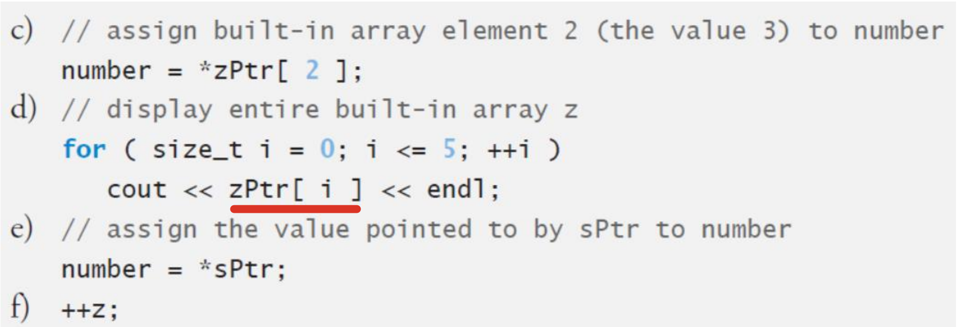
✓ b) A pointer that is declared to be of type void * can be dereferenced.

✓ c) A pointer of one type can't be assigned to one of another type without a cast operation.

Find the error in each of the following program segments. Assume the following declarations and statements:

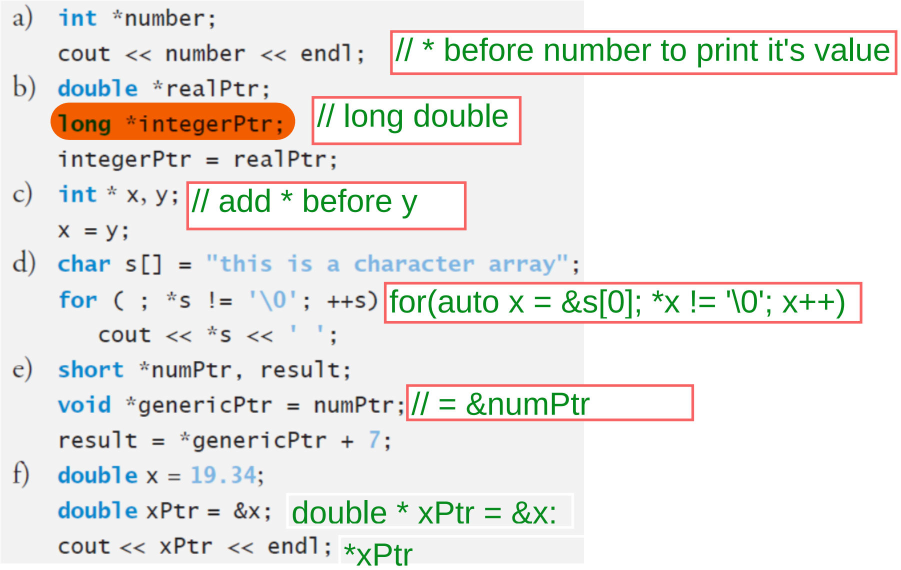


```
int *zPtr; // zPtr will reference built-in array z
void *sPtr = nullptr;
int number;
int z[ 5 ] = { 1, 2, 3, 4, 5 };

a) ++zPtr;
b) // use pointer to get first value of a built-in array
   number = zPtr;
```



```
c) // assign built-in array element 2 (the value 3) to number
   number = *zPtr[ 2 ];
d) // display entire built-in array z
   for ( size_t i = 0; i <= 5; ++i )
      cout << zPtr[ i ] << endl;
e) // assign the value pointed to by sPtr to number
   number = *sPtr;
f) ++z;
```

Find the error in each of the following segments. If the error can be corrected, explain how.



```
a)  int *number;
    cout << number << endl;  // * before number to print it's value
b)  double *realPtr;
    long *integerPtr;  // long double
    integerPtr = realPtr;
c)  int * x, y; // add * before y
    x = y;
d)  char s[] = "this is a character array";
    for ( ; *s != '\0'; ++s) for(auto x = &s[0]; *x != '\0'; x++)
       cout << *s << ' ';
e)  short *numPtr, result;
    void *genericPtr = numPtr; // = &numPtr
    result = *genericPtr + 7;
f)  double x = 19.34;
    double xPtr = &x;  double * xPtr = &x:
    cout << xPtr << endl; *xPtr
```