Chapter 2 - Flow of Control

These test questions are true-false, fill in the blank, multiple choice, and free form questions that may require code. The multiple choice questions may have more than one correct answer. You are required to mark and comment on correct answers.. Mark *all* of the correct answers for full credit. The *true false* questions require an explanation in addition to the *true*/*false* response, and, if *false*, also require a correction.

True False:

1. The if, while and for statements control only one statement. Answer:

   True

   Explanation: The one statement may be a block (statements that are enclosed with curly braces { }) or a simple statement.

2. Given the declaration

   ```
   int x = 0;
   ```

   The following expression causes a divide by zero error:

   ```
   (x !=0) ||        (2/x < 1);
   ```

   Answer: False.

   Explanation: The|| operator uses short-circuit evaluation. The first member of this expression is true; the truth value of the complete expression can be determined

from this; consequently, the second expression is not evaluated. There is no divide-by-zero error.

3. Suppose we have these declarations,

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

This Boolean expression is correct and it does what the programmer intends.

```
x < y < z
```

Answer: False

Explanation: Unfortunately, the expression compiles without error and runs. The < operator associates (groups) left to right, so the expression evaluates as

(x < y) < z

The left hand expression evaluates to true, which, if compared to a numeric type, converts to 1. When compared to 1, this is false. What the programmer intends, expressed as mathematacs might is -1 < 0< 1, a result that is clearly true.

4. You want to determine whether time has run out. The following code correctly implements this.

   !time > limit

   Answer: False.

   Explanation: The expression always evaluates to false. This cannot be what the programmer intended. The compiler doesn't catch the problem because the code is legal, correct C++. Corrected code is !(time > limit)

   Code execution proceeds as follows: The operator ! takes a bool argument. It returns the opposite bool value. The value of time is converted to a bool. The value of time is certainly nonzero, hence !time is !true, i.e., false. The > compares this result with a numeric value, limit, (an int or perhaps some kind of floating point). The value on the left (false) is converted to a 0 value of that type. The value of limit is unlikely to be a negative number and we are concerned about time running out, so it is unlikely that time is zero. Consequently, the inequality becomes 0>limit, where limit is nonzero. This is false.

5. The value of count is 0; limit is 10. Evaluate:

   (count == 0)&&(limit < 20)

   Answer: true

6. The value of count is 0; limit is 10. Evaluate: count

   == 0 && limit < 20

   Answer: true

Explanation: The operators == and < have higher precedences than &&, hence the expressions, count == 0 and limit < 10 are evaluated (to true) then the && is executed.

7. The value of count is 0; limit is 10. Evaluate:

   (count != 0)||(limit < 20)

   Answer: true.

   Explanation:Thefirstexpressionevaluatestofalse,thevalueofthe||expression is determined by the second expression. The second expression is true so the || expression evaluates to true.

8. In a while loop, the Boolean_Expression is executed before each execution of the loop body.

   Answer: true

9. In a do-while loop, a continue statement terminates the loop.

   Answer: False

   Explanation: The continue statement causes the Boolean_Expression to be executed. If true, the body executes, otherwise the loop terminates.

10. A break statement is used in loops only.

    Answer: False.

    Explanation: In addition to its use in loops, a break statement is used in the switch statement to transfer control to the next statement after the switch block.

11. When a loop is nested in side another loop, a break or continue statement terminates or restarts the outermost loop of the nested loop structure.

    Answer: False

    Explanation: A break or continue terminates or restarts only the innermost loop containing the break or continue.

Free Form Questions:

1. Assume variables first and second are declared to be double and are initialized.Writeasequenceoflinesofcodethatcausethevaluesstoredinfirst and second to be exchanged if the value of first is not less than second.

Answer:

```
//   double first, second;
//   these have been initialized if (!(first <
second))
{
      double temp = first; first =
      second; second = temp;


}
//assert: first <= second
```

2. Write multiway if-else statements for which the output is "Alarm: Boiler Pressure: TOO HIGH" if the value of the variable boiler_pressure is greater than 1000 (psi), and the output is "Boiler Pressure: TOO LOW" if the value of boiler_pressure is below 100(psi), otherwise the output is "Boiler Pressure: within normal limits."

Answer:

```
if (boiler_pressure > 1000)
    cout << "Boiler Pressure: TOO LOW\n";
else if (boiler_pressure < 100)
    cout << "Boiler Pressure: TOO LOW\n";
else
        cout << "Boiler Pressure: within normal      limits.\n";
```

3. Write multiway if-else statements in which letter grades are assigned based a numeric grade based on this "ten point" scheme:

    if the numeric grade is not less than 90, the letter grade is an A,

    if the numeric grade is not less than 80, the letter grade is a B,

if the numeric grade is not less than 70, the letter grade is C,

if the numeric grade is not less than 60, the letter grade is D,

otherwise the letter grade is F.

Answer:

```
if (numeric_grade >= 90)
      letter_grade =  'A';
else if (numeric_grade >= 80)
      letter_grade =  'B';
else if (numeric_grade >= 70)
      letter_grade =  'C';
else if (numeric_grade >= 60)
      letter_grade = 'D';
else
      letter_grade =          'F';
```

4. Assume variables first, second, and max are declared to be double and are initialized. Write a sequence of lines of code that cause the larger of the values in first and second to be stored in max.

   Answer:

```
//double first, second, max
//first and second have been initialized if ( (first < second)
)
      max = second;
else
      max = first;
//assert: max >= first && max >= second
```

5. A numeric integer grade is between 50 and 99. Using integer division or otherwise obtain a int value that is 5, 6, 7, 8, or 9 from the numeric grade. Write code using a

switch statement using this number in the selector expression that assigns letter grades based on this 10 point scheme:

if the numeric_grade is not less than 90, the letter_grade is an A,

if the numeric_grade is not less than 80, the letter_grade is a B,

if the numeric_grade is not less than 70, the letter_grade is C,

if the numeric_grade is not less than 60, the letter_grade is D,

otherwise the letter_grade is F.

Answer:

```
int value = numeric_grade/10;
switch(value)
{
    case 9: letter_grade = 'A';
            break;
    case 8: letter_grade = 'B';
            break;
    case 7: letter_grade = 'C';
            break;
    case 6: letter_grade = 'D';
            break;
    default: letter_grade = 'F';
            break;
}
```

6. Write Boolean expressions that represent the given English expressions. Assume any variables used have been declared and initialized.

    a) alpha is greater than 1

b) x is odd

c) x and y are odd

d) ch is an upper case alphabetic character (between 'A' and 'Z').

e) digit, which is f type char, has value that is indeed a digit.

a. alpha > 1

b. (x%2==1)

c. (x % 2==1) && (y % 2==1)

d. ('A' <= ch) && (ch <= 'Z')

e. ('0' <= digit) && (digit <= '9')

7. Write Boolean expressions that represent the given English expressions. Assume any variables used have been declared and initialized.

   a) at least one of x or y is odd

   b) at least one of x or y is non-negative (*x is non-negative* is written x >= 0)

   c) The hit is no more than 2.5 units away from target

   d) x is 1 or x is equal to y

   e) t is strictly between 3.2 and 3.3

   a) (x%2==1) || (y%2==1)

   b) (x >= 0) || (y>=0)

   c) ((hit - target) <= 2.5) && ((hit - target) <= -2.5)

   d) (1==x) || (x==y)

   e) (3.2 < t) && (t < 3.3)

12. Explain the programmer's saying from the text, section 2.2, "Garbage in means garbage out."

    Answer: This remark means that if you supply a program with bad data, you are guaranteed useless results from your program.

13. Use the condition operator (x?y:z) to write a very compact expression that assigns the maximum of variables n1 and n2 to the variable max. You should assume that any variables you use have been declared and initialized appropriately.

    Answer: max = (n1>n2) ? n1 : n2;

    Explanation: The parentheses are present only for readability. That the > operator has higher precedence makes the parentheses unnecessary.

14. What is the output from each of the following loops? a)

```
while ( 0 )
    cout << 'X';
cout << endl;
```

b)

```
do
    cout << 'X';
while ( y != y );
cout << endl;
```

c)

```
int i = 1; while (i
< 9)
{
    cout i;
    i++;
}
cout << endl;
```

d)

```
char c = 'A'; do

{
    cout << c << " "; c =
    c + 2;
} while ( c <= 'M' ) cout
<< endl;
```

e)

```
int i = 0; while (i <
50)
{
```

```
        if ( i < 20 && i != 15 )
            cout << 'X';
        i++;
    }
    cout << endl;
```

Answer:

The output of each loop is:

a. The only output is a carriage return.

b. X

c. 12345678

d. A C E G I K M

e. XXXXXXXXXXXXXXXXXXX (There are 19 Xs.)

15. Write the following do-while statement with a while construct, and maybe some extra code.

```
x  =  10;
do
{
        cout << x << endl;
        x = x - 3;
} while ( x > 0 );
```

Answer: A simple minded change from do while to while and insertion of the loop body gives this:

```
x =  10;
cout << x << endl;
x = x - 3;
while ( x > 0 )
{
   cout << x << endl;
   x = x - 3;
}
```

A look at the code suggests that the following is somehow

```
'smarter' x = 10;
while ( x > 0 )
{
    cout << x << endl;
    x = x - 3;
}
```

16. Write a program that reads in exactly 10 integers and outputs the sum. Answer:

```
#include  <iostream>
//loop to accept 10 integers and sum int main()

{
    using namespace std;
    int x, sum = 0;
    cout      << "Enter 10 integers, each followed by <cr>"
              <<   " I will give the sum." << endl; for(int i =0;
    i < 10; i++)
    {
        cout << "Enter integer " << i << ": "; cin >> x;

        sum = sum + x;
    }
    cout << "sum: " << sum << endl; return 0;

}
```

17. Write a program that reads in and sums the squares of positive integers until a value that 0 or less is read in.

Answer:

```
//   file  ch2nu36.cc
//   test question 36 for chapter 2
```

```
>>Delete above comment
#include <iostream>
using namespace  std;

//loop to accept positive integers until nonnegative //integer, then return
sum int main()

{
    int x = 1, i = 0, sum = 0;
    cout << "Enter positive integers, followed by <cr>"
         <<   " 0 or negative stops." << endl
         <<   " I will give the sum." << endl; while ( x >
    0 )
    {
        cout << "Enter integer " << i << ": "; cin >> x;

        sum = sum + x;
        i++;
    }
    cout << "sum: " << sum << endl; return 0;

}
```

18. For each of the following situations, tell which type loop (while, do-while, or for) would be best in that situation:
    a) Reading a list of an unknown number of homework grades for a single student.
    b) Summing a series such as $1 + 1/2 + 1/(2^2) + 1/(2^3) + \ldots + 1/(2^8)$
    c) Testing a newly coded library function to see how it performs for different values of its arguments.
    d) Reading in the number of days of vacation taken by an employee.

    Answer:

    a) A while loop because the list of grades may be empty.

b) A for loop, because the length of the list of numbers is known.

c) A do-while loop could be used since there will be at least one value tested.

d) A while loop because the list of grades may be empty.

20. Predict the output of the following nested

loops:

```
int n = 1;
while(n <= 10)
{
    int m = 10;
    while(m>=1)
    {
        cout << n << " times " << m
                << " = " << n*m << endl;
        m--;
    }
    n++;
}
```

Answer: The output is a multiplication table

```
1 times 10 = 10
1 times 9 = 9
. . .
1  times     1    =    1
2 times 10 =  20
. . .
2  times     1    =    2
3 times 10 =  30
. . .
3  times     1    =    3
4 times 10 =  40
4 times 9 = 36
. . .
```

```
4  times  2  =       8
4  times  1  =       4
. . .
9 times 10 = 90
. . .
9   times 1 = 9
10  times 10 = 100
. . .
10    times 1 =  10
```

21. Rewrite the following while loops as for loops:

a)
```
int i = 1;
while(i<=10)
{
        if (i<5 && i !=2)
           cout  << 'X';
    i++;
}
```

b)
```
int i =1;
while(i<=10)
{
    cout  << 'X';
    i = i + 3;
}
```

c)
```
int n = 100; do

{
      cout  <<  'X';  n
      = n + 100;
 }while(n <  1000);
```

Answer:

a) for(int i=1; i<=10; i++) if (i<5
&& i !=2)

cout << 'X';

b) for(int i=1; i<=10; i = i + 3) cout << 'X';

c) for(int n = 100; n <1000; n = n + 100) cout
<< 'X';

Multiple Choice

There may be more than one correct answer. You must give all correct answers for full credit. An explanation is required.

1. Which control construct repeats a sequence of statements zero or more times?
   a) while statement
   b) do-while statement
   c) for statement
   d) switch statement
   e) if-else statement

   Answer: a), c)

   Explanation:: b) repeats 1 or more times, d) and e) are selection statements
2. What is the value of the bool valued expression, 1 < x < 10? Does the value of this depend on the value of x? Explain, and give the expression that the programmer probably meant.
   a) This statement is incorrect as it is always false.
   b) This statement is correct and its value depends on x.
   c) This statement is incorrect and it is always true
   d) This statement is incorrect, but it does depend on the value of x.

   Answer: c) This expression is always true. The value does not depend on x. This is the mathematicians' shorthand for what the programmer probably meant:

   (1 < x)&&(x < 10)

Explanation: The < operator associates (groups) left to right, so the expression evaluates as (1 < x) < 10. The expression (1 < x) evaluates to either false or true, regardless of x. These bool values convert to int values 0 or 1 when compared to int value 10. The expression evaluates to true for all values of x.

3. Which of the following is true of the && operator?
   a) It has two operands.
   b) It can have one operand.
   c) It uses short circuit evaluation.
   d) It is the logical AND operator.
   e) It returns true if either operand is

   true. Answer: a), c), and d).

   Explanation: b) is a wrong number of argments, e) is the OR command.
4. Which of the following is true of the || operator?
   a) It has two operands.
   b) It can have one operand.
   c) It is the logical OR operator.
   d) It returns true if either operands is true.
   e) It uses short circuit evaluation.

5. In a switch statement, when a break statement is encountered, an immediate transfer of control is made to
   a) the default case of the switch statement
   b) a goto statement
   c) the else clause
   d) the statement beyond the end of the switchstatement.
   e) none of these

6. An assignment of the value of a conditional expression to a variable (x =y?z:w;) can always be replaced by

a) a switch statement with assignment statements for its case statements.
b) one or more ifs with else clauses and assignment statements for its true and false clauses.
c) one or more nested while loops with assignments for the bodies of the loops.
d) oneormoreifswithoutanyelseclausesandassignmentstatementsforits yes_statement(s).
e) none of these is correct.

Answer: a), b), c) and d)

Explanation: c) is correct too, though strange. Here is one solution:

```
#include<iostream>
using namespace std;
int  main()
{
   int x , z = 10, w =-10;
   bool y = true;
   while( y ||  (x = w))
   {
      while (y)
      {
           x = z;
           break;
      }
      break;
   }
   cout << x << endl;

   y = false;
   while( y ||  (x = w))
   {
      while  (y)
```

```
            {
                x = z;
                break;
            }
            break;
        }
        cout << x << endl;
        return      0;
    }
    /* Output is:
    10
    -10
    */
```

7. In distinguishing an expression as true orfalse, C++ sees which of the following as true?
   a) true
   b) 0
   c) 1
   d) Any non-zero value
   e) The character 'F'

   Answer: a), c), and d) e) ‘F’ is coerced char to bool, perhaps through an intermediate type
8. Which of the following determines the operator that is processed prior to another operator?
   a) Operator associativity
   b) Operator precedence
   c) Whether the operator is an arithmetic operator
   d) None of these determine the order in which operators are processed. Answer: b)

9. The following program purports to sum all entered int values that are greater than 5. It compiles without any error message, and it executes without error message, but nevertheless is wrong. Name all the errors.

```
//  Display the sum of all entered int values
//  greater than 5
#include <iostream>
int  main()
{
    using namespace std;
    int x, sum;
    while (x < 10)
    {
        cin >> x;
        if (x > 5);
            sum = sum +x;
    }
    cout << "The sum is values > 5 is " << sum << endl;
}
```

a) The while header needs a semicolon at the end of its line.

b) The semicolon at the end of the if statement is an error that the compiler should catch.

c) The semicolon at the end of the if statement causes all entered values to be summed.

d) The sum variable is not initialized, so the output value is most likely garbage.

Answer: c) and d) are the errors. (Perhaps the user should have been prompted.)

10. Which of the following loop statements is guaranteed to iterate the body of the loop at least once?

a) while(control) body;

b) do body while(control);

c) for (initialize; test; update) body; d) none of the above

e) all of the

above Answer: b)

11. A switch statement must have

a) a default case

b) more than one non-default case

c) a break statement

d) none of the above

e) all of the above

12. An enumeration type

a) is type whose values are defined by a list of constants of type int.

b) is a type separate from any of the integer types

c) can be converted to integer types

d) is a type that a programmer should avoid doing arithmetic

with. Answer: all, a), b), c), and d), are correct.

13. The comma operator

a) is a list of expressions separated by commas

b) according to the ANSI C++ Standard, is supposed to be evaluated left to right

c) not all compilers evaluate left to right, i.e., do not follow the C++ Standard, hence left to right evaluation should not be depended upon.

d) has value equal to the value of the first expression in the list.

e) all of the above

Answer: a) b) c).

Explanation: d) is wrong, the correct statement is: A comma expression has value equal to the value of the *last* expression in the list.

14. Where is it legal to put a continue statement? What does the continue statement do there?

a) A continue statement causes an unnested loop torestart.

b) A continue statement causes a loop to halt.

c) A continue statement in a loop nested in another loop causes the entire nested loop to restart.

d) A continue statement in switchstatement transfers control to the top of the switch.

e) A continue statement in a nested loop causes that loop to restart, there is no effect on other loops.

Answer: a) e) are correct.

14. Where is it legal to put a break statement? What does the break do there?

a) A break is placed in a simple (unnested) loop, to terminate the loop.

b) A break is placed in an inner block of nested blocks, to transfer control beyond the end of block the break is within.

c) A break is placed in a loop in nested loops, to transfer control beyond the end of the innermost loop the break is within.

d) A break is placed in a switch statement, to terminate the switch by transferring control beyond the end of the switch.

e) A break is placed in a loop where it restarts the loop.

15. If this code fragment were executed in an otherwise correct and complete program, what would the output be? Explain.

```
int a = 3, b = 2, c = 5 if (a >
b)
    a = 4;
    if ( b > c) a =
        5;
else
    a = 6;
cout << a < endl;
```

a) 3

b) 4

c) 5

d) 6

e) None of the above, the cout statement belongs to the else and so is skipped.

Answer: d)

Explanation: Theelsebelongsto thesecondif,despiteindentationsuggesting otherwise. Consequently, the first if statement executes, a is assigned the value 4, the second if executes, b is 2, c is 5, so b > c, evaluates to false so the else clause executes, so that a is assigned the value 6.

16. Here is a collection of if and if-else statements with semicolons in various places. Assume all variables have been declared and initialized. Which of these are correct and are likely to give the programmers intent? Which are correct but unlikely to give the programmer's intent? Give the error for the remaining.

a)
```
if ( a > b ); a =
        b;
else
        b = a;
```

b)
```
if(a > b ) a =
                b;
else;
        b = a;
```

c)
```
if(a > b )
        a = b;
else
        b = a;
```

d)
```
if(a > b)
      a = b
else
        b = a;
```

e)
```
if( x !=0 ) a =
                a / x
```

Answer: c) is correct and is likely to be the programmer’s intent. b) compiles but is unlikely to be the programmer’s intent.

Explanation:

a) Compiler error: The semicolon at the end of the if line introduces a null statement, making the else keyword an error. The error will not be detected by the compiler until the else is encountered.

b) Compiles with no errors, but is unlikely to do what the code author intended: The indentation suggests that the programmer meant the a=b and b=a lines to be alternatives chosen based on whether a>b. The semicolon at the end of the else causes the statement that appears to be the else clause always to be executed.

c) correct, and apparently does what the programmer intended.

d) Here the compiler will find an error at (or after) the else, since this is the first token after the assignment a=b, where a semicolon is needed.

e) This is the same error as in d), but the error will be detected at or after the last x on the second line.

17. Here is a collection of while and do-while statements. Identify:

i. those that are correct, and are likely to give the programmers intent;

ii. those that are correct, but unlikely to give the programmer's intent, and

iii. what compiler error will the rest generate?

Assume all variables have been declared, but not necessarily initialized.

a)
```
cin >> n; while (-1
!= n)
{
    sum = 0;
    sum = sum + n;
}
```

b)
```
cin >>  value;
while ( value != -1 ) sum =
    sum + value;
cin >> value;
```

c)
```
cin >> n; int i = 1,
>>Semicolon not comma
while ( i < n );
```

```
        sum = sum + i;
    i++;
```

d)
```
cin >> count >> limit; do

    count++
while ( count * count > limit );
```

e)
```
cin >> x;
do
    x++; while( x
> x );
```

a) This compiles. It is unlikely to be what the programmer intended. What the intent might be is hard to guess, since the value of sum is reset to 0 each time the loop executes.

b) This compiles. The indentation suggests that the programmer intended the two lines following the to be in the body of the loop. The second of these is not controlled by the while clause, resulting in an infinite loop .
>>Something is wrong. Maybe the indentataion??

c) This compiles. There are two intent errors evident: the semicolon on the second line and the missing braces. If the loop is entered, this is an infinite loop, since the statement controlled by the loop is the null statement inserted by the semicolon on the second line changes neither i nor n. The intent evidently was to run the loop n or n-1 times.

d) The syntax error is a semicolon missing from count++.

e) This compiles, but the loop executes its body only once. It is difficult to guess what the programmers intent might have been, but this probably isn't it!

18. If the following code fragment is executed in an otherwise complete and correct program, which expression will be executed? Why?

```
x = 0;
if (x = 12)
   yes_statement;
```

```
else
    no_statement;
```

a) The no_statement will be executed because x is not 12.
b) The statement has incorrect syntax so will not compile at all.
c) x=12 is illegal in the Boolean expression of an if statement.
d) The yes_statement will be executed.

Answer: d)

Explanation: The expression x = 12 has the value 12, which is converted to true, causing the if always to execute the yes_statement.

19. Which of the following control structures *requires* curly braces?

a) if-else
b) while
c) do-while
d) switch
e) for

Answer: d)

Explanation: The other control constructs operate on a single statement, which may be, but *is not required to be*, a compound statement.

20. In the expression (j > 0 && j+1 == 10), which operator executes *last*?

a) >
b) &&
c) + d) ==

Answer: b)

Explanation: The precedences, higher to lower are: +, >, ==, &&. So the order is j**+**1, j**>**0, then (j+1)**==**10, then (finally) the ((j>0) **&&** ((j+1) == 10))

21. When you don't recall operator precedences you can

a) Look in a table of precedences
b) Guess
c) Use parentheses

d) Experiment with the compiler

Answer: c) is perhaps best and easiest, but a) and d) are reasonable if you aren't taking an exam.

22. Consider the if statement:

if(condition) yes_clause; else no_clause;

Under which of the following circumstances will both the yes_clause and the no_clause will be executed?

a) When the condition is true.
b) When the condition is false.
c) When the condition is ambiguous.
d) This will not happen.

23. The following are true about the expression left && right.

a) The expression is false when left is false and right is false
b) The expression is true when left is true and right is false
c) The expression is false when left is false and right is true
d) The expression is true when left is true and right is true Answer: a) c) and d)

24. The following are true about the expression left || right.

a) The expression is false when left is false and right is false
b) The expression is true when left is true and right is false
c) The expression is false when left is false and right is true
d) The expression is true when left is true and right is true Answer: a) b) and d)

25. The statements int x = 1; int y; y = x++;

a) Assign y the value 2;
b) Change the value of x to 2.
c) Assign y the value 0;
d) Assign y the value 1;
e) The ++ is a postfix operator.

Answer: b) d) and e)

26. The statements int x = 1; int y; y = --x;
    a) Assign y the value 1;
    b) Change the value of x to 0
    c) Assign to y the value 1;
    d) Assign to y the value 0;
    e) The -- is a prefix operator.

27. What is the output of the following, if it were embedded in an otherwise correct and complete program and run?

```
int x = 10; while
(x > 0)
{
        cout << x << " ";
        x = x + 3;
}
cout << endl;
```

    a) 10 13 16 19 . . .
    b) The compiler detects that this will be an infinite loop, so it does not compile. Insert lowercase be
    c) This is an infinite loop. When compiled and run, it runs until machine limitations stop it, or you get tired of it and kill the process.
    d) 0 3 6 9.

28. This question asks about nesting of if, if-else, switch, while, do-while, and for statements:
    a) These constructs may not be nested in at all.
    b) These constructs may be nested in any way that meets the needs of algorithms the programmer is coding.
    c) Only control constructs of a given kind may be nested (while loops within while loops; if-else within if-else etc.)

d) The question does not make sense in C++.

Answer: b) Control constructs may be nested arbitrarily.

29. Each of the following has at least one error, either intent, or an error that may be caught by the compiler or both). What is the error? Assume all the variables you see are defined and initialized.

a)
```
for(int i = 0; i <10; i++); sum =
    sum +x;
```

b)
```
if (x > 0)
    x  = 3
else
    x   =4;
```

c)
```
if(x = 0) x = MAXINT;
```

d)
```
if x > 0 x = 3;
```

e)
```
if (x>0) then x  =3;
```

Answer: a) Not wrong, but the semicolon is probably an intent error. b) Semicolon missing in the yes_clause. c) The = probably should be ==. d) needs parenthese to be correct C++. e) C++ does not use “then” as a keyword. If we blindly fix the syntax, whatever the ”then” might be, there needs to be a semicolon after it, and the sequence then; x=3; should probably be in curly braces. Actually, there no way to guess what the programmer might mean here. There is a good chance the programmer is a refugee from Pascal, and then should be deleted.