**Review Questions for C++ Programming language**

**Control statements + functions + Arrays**

Given a program as follows, what would be printed?

#include <iostream>

using namespace std;

int myfunc(int);

int myfunc(float);

int main(){

cout << myfunc(3) << "\n";

return 0;}

int myfunc(int n){

return n \* 2;}

int myfunc(float n){

return n \* 3.0;}

a- 7 b- 7.02 **c- 6** d- 10.53

#include <iostream>

using namespace std;

int sub1(int &n){

n--;

return n;}

int main(){

int m = 10;

int j=5;

m -= sub1(j);

cout << m << "\n";

return 0;}

a- 0 b- 1 c- 6 d-10

#include <iostream>

using namespace std;

int f(int &i){

i = 10;

return(5 \* i);}

int main(){

int n = 5;

cout << f(n) <<"-";

cout << n << "\n";

return 0;}

a- 50-10 b-10-50 c- 50-50 d- 15-10

#include <iostream>

using namespace std;

double sum(int pt[], int n){

int temp = 0;

for (int i = 0; i < n; ++i) {

temp += pt[i];}

return temp;}

int main(){

int total;

int pt[5];

for (int i = 0; i < 5; i++)

pt[i] = i;

total = sum(pt, 3);

cout << total << endl; return 0;}

a- 3 5 b- 3 c- 6 5 d- 10 3

#include <iostream>

using namespace std;

int fun1();

int fun2(int);

int fun3(int , int);

int fun4(int &);

int main(){

int i = 5;

cout << fun1() ;

cout << fun2(3) ;

cout << fun3(6,2) ;

cout << fun4(i) ;

return 0;}

int fun1( ){

int y = 3 ;

return y\*y;}

int fun2(int x ){

return x+x ;}

int fun3(int x , int y){

return x / y ;}

int fun4(int &x ){

return x-x ;}

a-9635 b- 6930 **c- 9630** d- 6903

int x = 0 \* (1 / 3 \* 3.0);

cout << x << endl;

**a- 0** b- 0.0 c- 1 d-1.0

{

int i = 1;

double x = 1.111;

cout << i << " " << x << "\n";

{

int x = 2;

double i = 2.222;

cout << i << " " << x << "\n";

}

}

**a-1 1.111 2.222 2** b-1.111 1 2.222 2 c-2.222 2 1 1.111 d-1.111 1 2 2.222

**Find Errors**

1-

int y=0;

while (y>0) {

cout<<"nice day";

y++;}

cout<<"Good bye";

**answer :** logical error :while will never entered (always false)

2-

int y=1;

while (y>0)

{cout<<"nice day";

y++;}

cout<<"Good bye";

**answer :** logical error : infinite loop

3-

int counter = 0;

while (counter <= 10)

cout<<"counter= "<<counter;

**answer :** Logical error (Infinite loop), since there is no update for the counter

4-

for (int counter=1; counter<=5; counter++)

cout<<counter<<endl;

cout<<counter+1<<endl;

**answer :** Syntax error counter is declared in the for and cannot be used outside its body.

Which of these function calls correct for the functions bellow ( more then one )

1. double GetHours(double Mon, double Tue , double Wed, double Thu , double Fri) {

double TotalHours = Mon + Tue + Wed + Thu + Fri;

return TotalHours;

}

1. **GetHours( 3.5,1.5,2.5,1.5,4.5)**
2. GetHours()
3. GetHours(‘a’,’b’,’c’,’d’,4)
4. **GetHours(3,4,4,5,1.5)**
5. GetHours(3.5,4,’a’,true,1.5)
6. **Int sum ( int x , int y )**

**{**

**return x + y ;**

**}**

1. **sum( 5 , 7 )**
2. sum( 6.3 )
3. **sum( 7 , 3.5 , ‘a’ )**
4. **sum( )**

**What is the output of the following code?**

**The output:**

1 2 4 5 7 8 10

for (int counter=1; counter<=10; counter++)

      { if (counter % 3 == 0)

      continue;

        cout<<counter<<" ";

      }

#include<iostream>

using std:: cout;

**The output:**

&&&

&&&

&&&

&&&

\*\*\*\*\*\*

using std:: endl;

int main()

{int x=3,y=4;

for (int i=1; i<=y; i++)

{for (int j=1; j<=x; j++)

cout<<'&';

cout<<endl; }

for (int i=1; i<=x; i++)

for (int j=1; j<=i; j++)

cout<<'\*';

return 0;}

void main( )

**The output:**

a= 2, b= 3

a=3, b=4

a=4, b=5

{

int a,b;

a=2, b=0;

while (a\*b<=24)

{

b=a+1;

cout<<"a= "<<a<<", b= "<<b<<'\n';

++a;

b+=2;

}}

#include <iostream>

using namespace std;

int main(){

int n1=10, n2, n3;

n2=n1+10; n3=n1+20;

for(; n1<n3 ;n1+=2)

{

**The output:**

10 12 14 16 18 22 26 28

if(n1==n2 || n1==(n2+4) )

continue;

cout<<n1<<"\t";

} return 0;}

#include <iostream>

using namespace std ;

double average(double , double , double );

int main()

{

double number1 = 4 , number2 = 5 , number3 = 7 ;

double average\_result = average(number1, number2, number3);

cout<<"the average of three numbers you entered is "<<average\_result;

return 0;

}

double average(double num1, double num2, double num3)

{

double avg;

avg = (num1 + num2 + num3) / 3;

return avg;

**The output:**

the average of three numbers you entered is 5.333

}

Write a function named "digit\_name" that takes an integer argument in the range from 1 to 9 ,inclusive, and prints the English name for that integer on the computer screen. No newline character should be sent to the screen following the digit name. The function should not return a value. The cursor should remain on the same line as the name that has been printed. If the argument is not in the required range, then the function should print "digit error" without the quotation marks but followed by the newline character. Thus, for example,

the statement digit\_name(7); should print seven on the screen;

the statement digit\_name(0); should print digit error on the screen and place

the cursor at the beginning of the next line.

**Answer:**

Void digit\_name(int digit\_value)

{

switch (digit\_value)

{

case 1 : cout << "one"; break;

case 2 : cout << "two"; break;

case 3 : cout << "three"; break;

case 4 : cout << "four"; break;

case 5 : cout << "five"; break;

case 6 : cout << "six"; break;

case 7 : cout << "seven"; break;

case 8 : cout << "eight"; break;

case 9 : cout << "nine"; break;

default : cout << "digit error" << endl;

}

Write a function named "sum\_from\_to" that takes two integer arguments, call them "first" and"last", and returns as its value the sum of all the integers between first and last inclusive. Thus, for example,

cout << sum\_from\_to(4,7) << endl; // will print 22 because 4+5+6+7 = 22

cout << sum\_from\_to(-3,1) <<endl;//will print 5 'cause (-3)+(-2)+(-1)+0+1 = 5

cout << sum\_from\_to(7,4) << endl; // will print 22 because 7+6+5+4 = 22

**Answer:**

int sum\_from\_to (int first, int last)

{

int i, partial\_sum = 0;

if (first <= last)

for (i = first; i <= last; ++i)

partial\_sum += i;

else

for (i = first; i >= last; --i)

partial\_sum += i;

return partial\_sum;}

Write a function named "enough" that takes one integer argument, call it "goal" and returns as its value the smallest *positive* integer n for which 1+2+3+. . . +n is at least equal to goal . Thus, for example,

cout <<enough(9)<<endl; // will print 4 because 1+2+3+4 \_ 9 but 1+2+3<9

cout <<enough(21)<<endl;// will print 6 'cause 1+2+ . . .+6 \_ 21 but 1+2+ . . . 5<21

cout <<enough(-7)<<endl;//will print 1 because 1 \_ 7 and 1 is the smallest

**Answer:**

int enough (int goal)

{

int n = 1, sum = 1;

while (sum < goal)

sum += ++n; return n; }

A positive integer n is said to be ***prime*** (or, "a prime") ( العدد الأولي ) if and only if n is *greater than* 1 and is divisible only by 1 and n . For example, the integers 17 and 29 are prime, but 1 and 38 are not prime. Write a function named "is\_prime" that takes a *positive* integer argument and returns as its value the integer 1 if the argument is prime and returns the integer 0 otherwise. Thus, for example,

cout << is\_prime(19) << endl; // will print 1

cout << is\_prime(1) << endl; // will print 0

cout << is\_prime(51) << endl; // will print 0

**Answer :**

int is\_prime (int n)

{

if (n <= 1)

return 0; // n cannot be prime if n <= 1.

int trial\_divisor = 2;

while (trial\_divisor < n && n % trial\_divisor != 0)

++trial\_divisor;

if (trial\_divisor == n) // n must be prime

return 1;

else

return 0; // n is not prime.

}

Write a function named "sum" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells how many floating point values are in the array.

The function should return as its value the sum of the floating point values in the array. Thus, for example, if the array that's passed to the function looks like this:

5.8 | 2.6 | 9.0 | 3.4 | 7.1

The function should return 27.9 as its value .

**Answer:**

float sum (const float a[], int n)

{

float sum\_so\_far = 0.0;

int i;

for (i = 0; i < n; ++i)

sum\_so\_far += a[i];

return sum\_so\_far;

}

Write a function named "subtotal" takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells the number of cells in the array.

The function should replace the contents of each cell with the sum of the contents of all the cells in the original array from the left end to the cell in question. Thus, for example, if the array passed to the function looks like this:

5.8 | 2.6 | 9.1 | 3.4 | 7.0

then when the function returns, the array will have been changed so that it looks like this:

5.8 | 8.4 | 17.5 | 20.9 | 27.9

because 5.8 + 2.6 = 8.4 and 5.8 + 2.6 + 9.1 = 17.5 and so on.

Note that the contents of cell 0 are not changed. The function should not return a value.

**Answer:**

void subtotal (float a[], int n)

{

int i;

for (i = 1; i < n; ++i)

a[i] += a[i-1];

}

Write a function named "location\_of\_largest" that takes as its arguments the following:

(1) an array of integer values;

(2) an integer that tells how many integer values are in the array.

The function should return as its value the subscript of the cell containing the largest of the values in the array.Thus, for example, if the array that's passed to the function looks like this:

58 | 26 | 90 | 34 | 71

then the function should return the integer 2 as its value. If there is more than one cell containing the largest of the values in the array, then the function should return the *smallest* of the subscripts of the cells containing the largest values.

**Answer:**

int location\_of\_largest (const int a[], int n)

{

int best = 0; // Location of the largest so far.

int i;

for (i = 1; i < n; ++i) // Start comparing at the second cell.

if (a[i] > a[best])

best = i;

return best;}

Write a function named "rotate\_right" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells the number of cells in the array;

The function should shift the contents of each cell one place to the right, except for the contents of the last cell,which should be moved into the cell with subscript 0 . Thus, for example, if the array passed to the function looks like this:

5.8 | 2.6 | 9.1 | 3.4 | 7.0

then when the function returns, the array will have been changed so that it looks like this:

7.0 | 5.8 | 2.6 | 9.1 | 3.4

Note: the function should not return a value.

**Answer:**

void rotate\_right (float a[], int n)

{

float temp = a[n-1]; // Hold the contents of the last cell.

int i;

for (i = n - 1; i >= 1; --i)

a[i] = a[i-1];

a[0] = temp;}

write a program that calculates the squares and cubes of the integers from 0 to n and uses

tabs to print the following neatly formatted table of values:

#include <iostream>

#include <cmath> // to use pow function

using namespace std ;

int main()

{

int counter= 0;

cout<<"Integer\tSquare\tCube\n";

while( ++counter <= 6)

cout << counter <<"\t"<<pow(counter,2.0)<<"\t"<<pow(counter,3.0))<<"\n";

cout<<"counter="<<counter;

return 0;

**The output:**

**Integer Square Cube**

**1 1 1**

**2 4 8**

**3 9 27**

**4 16 64**

**5 25 125**

**6 36 216**

**counter=7**

}