OL Academy

# ITCS110/CSC103

First Semester, 2025-2026

**Final Exam Revision**



## Question (1) : MCQs

1. One of the following statements must be used to correctly compile and run a program dealing with files:

(a) # include <iostream>
(b) # include <fstream>
(c) #include <iomanip>
(d) # include<cmath>

2. The void function returns

(a) int
(b) double
(c) char
(d) nothing

3. The statement used to close an opened file named **inf** is

(a) inf.close();
(b) inf.closeFile();
(c) Close("info");
(d) closeFile("inf");

4. The correct header of the function max that finds and returns the large value in an array of the integers is

(a) int max (int a[], int size)
(b) void max (int a[size])
(c) void max (int a[], int&size)
(d) int max (int a, int size)

5. The C++ statement/s that define/s a file myfile.txt with a name file1 ar open/s it for reading is/are

(a) ifstream file1;
file1.open("myfile.txt");
(b) ifstream file1.txt;
file1.open();
(c) ifstream file1;
file1.open();
(d) ifstream file1 (myfile.txt);

6. The output produced after executing the following code is

```
char t[] = {'F', 'U', 'N'};
for (int i=0; i<3; i++)
    cout<< t[i] << '-';
```

(a) FUN
(b) 'F'-'U'-'N'-
(c) F-U-N-
(d) FUN---

7. The statement used to call a function with the header : void FUN (int x[], char&t) is

(a) cout<< FUN(x,t);
(b) cout<< FUN(x,t);
(c) FUN(x,t);
(d) cout<< FUN(x[],t);

8. The statement that defines a 2x2 matrix and initializes it with first row=20,-30; second rows = -9,88 is

(a) int M[][2] = {20,-30,-9,88};
(b) int M[][2] = {{20,-30},
(c) int M[2x2] = {{20,-30},{-9,88}};
(d) int M[2,2] = {{20,-30},

9. The following loop will be repeated -------- times.

```
for (int j=10; j>0; j++)
    cout<< 2*j << '\t';
```

(a) infinite
(b) 0
(c) 10
(d) 11

10. The output produced after executing the following code is

```
int x = 10;
for (int i=5; i<7; i++)
    for (int k=8; k>=7; k--)
        x+= k – 2*i;
cout<< x<< endl;
```

(a) -4
(b) 131
(c) 10
(d) -2

## Question (2) : OUTPUT

**(1)** Show the output of the following code

```
int y = 8, x = 25;
cout <<"X="<< x / 5 << endl;
switch (x / 5)
{
case 1: y++;
case 2:
case 3:  y = y + 2;
        break;
case 5:
case 7:  y = y % 3;
}
cout <<"Y="<< y << endl;
```

**(2)** What is the output of the following code:

```
bool w=false;
int x=10;
while (!w){
      x+=10;
      if (x<=30)
            continue;
      else if (x>=60)
            break;
      else
            cout<<"x="<<x<<endl;
}
```

**(3)** What is the output of the following code:

```
    for (int i=1; i<5; i++)
    {
        cout<<i<<" ";
        for (int j=i; j>1; j--)
        {
            cout<<j<<" ";
            if (i+j>4)
                continue;
            cout<<i+j<<" ";
        }
        cout<<endl;
    }
```

**(4)** What is the output of the following code:

```
void calcF(){
      static int y=5;
      int x=2;
      y*=x;
      ++x;
      cout<<x<<"\t"<<y<<endl;
}

int main(){
      for(int i=1;i<=2;++i)
            calcF();
     return 0;}
```

**(5)** What is the output of the following code:

```
int x=30;
void func(int& b, int z)
{
      int  x=50;
      cout<<"In function"<<endl;
      cout<<"x="<<x+3<<endl;
      b = b/ 2.0;
      z += 3;
      cout<<"z="<< z <<" b="<<b/
2.0<<endl;
}
int main(){
      int a=5;
      func(a, x);
      cout<<"a="<<a<<" x="<< x<<endl;
return 0;
```

**(6)** What is the output of the following code:

```
void check(int a, float b)
{
static float k = 5.0;
k += a + b;
int m = (a + 1) % 3;
cout<< k <<"\t"<< m <<endl;
}
int main()
{
int h = 4;
check(h, 9.0);
check(h, 5.0);
return 0; }
```

**(7)** What is the output of the following code:

```
void NumRef(double & P1, long & P2)
{
string str = " >> ";
P1++;
cout<< P1 << str << P2 << endl;
P1++;
P2 = P2 % 5 + 1 ;
}

int main() {
double C=2.1;
long D=7;
NumRef(C, D );
cout<< C <<"\t"<< D <<endl;
return 0;
}
```

**(8)** What is the output of the following code:

```
int num[6]={8,12,17,21,13,5}, x;
for (x=0; x<3; ++x){
     if (num[x]>num[6-x-1])
          cout<<num[x]<<endl;
     else
          cout<<num[6-x-1]<<endl;
}
```

**(9)** What is the output of the following code:

```
int a[3][4]={{4, 6, 4},{8,2}, {9, 3, 1,
1}}, sum=0;
for (int i=1; i<=2; i++)
{
    for(int j=0;j<=2;j++)
        sum+=a[j][i];

    cout<<i<<"\t"<<sum<<endl;
}
```

**(10)** What is the output of the following code:

```
char arr[3][3]={ {'A','H','B'},
{'T','K','N'}, {'D','I','G'} };

for (int k=1; k<=2; k++)
     for(int g=1;g<=2;g++)
          cout<<arr[g][k];
```

**Question (3):**

Write a C++ program that uses nested for-loops to display a rectangle. The program should ask the user to enter the number of rows and the number of columns. The number of rows should be between 1 and 100 inclusive. The number of columns should be between 1 and 50 inclusive. Appropriate error message should be displayed if the user inputs are invalid,

The rectangle should consist of * and # as in the following example:
If the number of rows is 5 and the number of columns is 9 then the rectangle should be:

| **Sample Input/ Output (1)** | **Sample Input/ Output (2)** |
|---|---|
| Enter number of row: 101<br>Enter number of col: 10<br>Invalid inputs | Enter number of row: 5<br>Enter number of col: 9<br>*########<br>#*#######<br>##*######<br>###*#####<br>####*#### |

**Question (4):**

**Part A:** Write a function that accepts two integer values as parameters, calculates and returns their sum or product. The function returns the sum of the parameters if they have similar signs, otherwise returns their product.

**Part B:** Write a main function that:

- Defines an array of 1024 integer elements, fill it from the keyboard.
- Calls the function defined in part a, for every pair of neighboring elements in arr, and stores the returned values in a new array of a proper type and size as shown below. Define this new array.
- Prints all elements of the new array separated by tab.

| Original array | 10 | 8 | 6 | -2 | ---------- | 12 | -2 | -3 | -12 |
|---|---|---|---|---|---|---|---|---|---|

| New array | 18 | -12 | ---------- | -24 | -15 |
|---|---|---|---|---|---|

```
int main ()
{
```

**Question (5):**

A) Write a C++ function named **findMax** that takes two parameters: a 2-dimensional array **temp** consisting of 7 columns and any number of **rowSize**. Rows represent weeks and columns represent days as shown below.

*Days*

| *Weeks* | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 22 | 22.5 | 22.6 | 22.8 | 23 | 23.2 | 22.5 |
| | 22.4 | 22.3 | 22.4 | 22.5 | 22.8 | 23.1 | 22.9 |
| | … | … | … | … | … | … | … |
| | … | … | … | … | … | … | … |
| | 22.3 | 24.1 | 23.1 | 23.2 | 23.3 | 22.9 | 22.8 |
| | 23.1 | 23.3 | 23.5 | 23.4 | 22.9 | 22.8 | 22.4 |

The function should find and print the week number and day number of the **hottest** day.

B) for the following data definitions, write one statement to call the above defined function to find and print the hottest day.

**double tempX[ ][ ] = {{20.0,21.3,..},...., {22.9,31.2,...}}; // number of rows is 12**

**Question (6):**

A file named "**Fruit.txt**" contains unknown number of fruit (maximum number of fruit is 50). Each line in the file contains fruit name (string), weight in kilograms (int) and price per kilogram (float). Write a C++ program that reads the data into 3 parallel arrays. Your program should display the following:

1. the number of fruit in the file
2. the amount of price for each fruit in the file

   **(Amount = weight in kilograms * price per kilogram)**
3. the highest amount of price with the name of the fruit

Format the amount of price to 3 decimal places. Sample Input/output of the program is shown in the example below:

**Fruit.txt (Input file)**

```
Apple        13    0.800
Banana       25    0.650
Mangos       10    1.050
....
Strawberry   4     0.900
```

**Screen Output**

```
There are 11 fruit in the file.

Fruit        Amount
Apple        10.400
Banana       16.250
Mangos       10.500
....
Strawberry   3.600

Banana has the highest amount of price = 16.250
```

## More Exercise:

**(7)** Write a C++ program that uses nested for-loops to display an **uppercase letter E of stars (*)**. The program should ask the user to enter an **odd positive integer** number (**N**) greater than or equal to 5 . Then the program should display the letter E of stars that consists of N rows and N columns. Below is 2 samples output based on N value as follows:

| If **N = 5**, then the output should look as follows: | If **N = 9**, then the output should look as follows: |
|---|---|
| *****<br>*<br>*****<br>*<br>***** | *********<br>*<br>*<br>*<br>*********<br>*<br>*<br>*<br>********* |

**Appropriate error message should be displayed if the input is not valid.**

**(8)**

**Part (A)** Write a function called ***Admission*** that accepts three parameters: School Score ***SS*** (*double*), Written Test Score ***TS*** (*double*) and Interview Score ***IS*** (*double*). The function will calculate total score as follows:

$$total\ \ score = SS + TS + IS$$

Then the function will return ***admitted college (****string****)*** as follows:

| Total Score | Admitted college |
|---|---|
| ≥ 90 | Engineering |
| 80 to 89 | IT |
| < 80 | Others |

**Part (B)** Complete the below main program to prompt the user to enter the School score (*double*), the Written Test Score (*double*) and the Interview Score (*double*). The program will then display the admitted college by calling the function defined in Part (A). Follow the sample below.

**SAMPLE INPUT/OUTPUT**

```
Enter School, Written test and Interview Scores:
65 12 14
Admitted College is Engineering
```

**(9)** Write a function called changeLines that accepts three parameters: a two-dimensional array of inegers called **matrix**, its row size **rowsize,** and its column size **colSize**. The maximum number of colums is 4.

The function swaps the integers of the first row with the integers of the last row, then prints the two dimensional array after swapping.

**Example:**

**Before calling a function**

| | | | |
|---|---|---|---|
| **5** | **8** | **9** | **2** |
| 8 | 9 | 1 | 5 |
| 4 | 8 | 3 | 7 |
| **2** | **5** | **9** | **1** |

**After calling a function**

| | | | |
|---|---|---|---|
| **2** | **5** | **9** | **1** |
| 8 | 9 | 1 | 5 |
| 4 | 8 | 3 | 7 |
| **5** | **8** | **9** | **2** |

**(10) Part (A)** Write a function called **average** that accepts two parameters, a 1D array marks, and the second parameter is the number of students. The function calculate and return the averages of the marks.

**Part (B):** Complete the main program to prompt the user to enter number of students, then create an array named marks and save them in the array marks.

| ***Sample Input/output*** |
| --- |
| `Please Enter the numeber of students (n): 4`<br>`Please Enter student marks: 9.5 1.5 3.5 10`<br>`The Average is = 6.125` |

**(11)** One week temperatures data has been recorded for 6 different cities. Write a function, named **findFreezingCities()**, that takes as parameter a two-dimensional array named, **temperatures** of size 6 x 7, that contains cities' temperatures in which each row represents a city and each column represents one-day temperature. The function should find and display the weekly average temperature for each freezing city. A city is considered freezing if the temperature is below 0 for at least four days in one week. Format the weekly average to 2 decimal places.

**For example**, if the array **temperatures**, contains the following data:

*Days Temperatures*

| *Cities* | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 5 | 11 | -1 | -3 | 0 | 2 | -2 |
| 1 | 7 | -4 | 0 | -2 | 1 | -5 | -4 |
| 2 | -2 | -5 | -5 | -4 | 3 | 1 | -1 |
| 3 | 7 | -4 | 0 | -2 | 1 | 5 | 12 |
| 4 | -1 | -1 | -1 | -2 | 3 | 3 | -2 |
| 5 | 3 | 5 | 1 | 1 | 0 | -3 | 2 |

Then the output of the function should be displayed as follows (note that highlighted rows indicate freezing cities.):

```
Freezing Cities Weekly Averages:
City#           Average
2               -1.00
3               -1.85
5               -0.14
```