[ANSWER SHEET]

[Module: Bscs 8th semester]

NAME: FARHAN KHAN   ID:13004

[farhankhan.mk199@gmail.com]

Name: FARHAN KHAN--Id:13004

Q1,a:

ANS:

If statement:

The if statement is used to check a condition and if the condition is true, we run a block of statements (called the if-block), and if the condition is false that block is ignored and the next statement in order is performed.

There is single entry and single exist in if statement.

Pseudocode example:

```
⇨ If the student's grade is greater than or equal to 60
          Print "Passed"
```

C++ code:

```
If(grade>=60)
        Cout<<"Passed";
```

Two forms of if statement are:

- **Else-if:**
  We use this statement (form) when we have multiple conditions for a problem.
  IF statement uses the keyword ELSEIF to introduce additional boolean expressions. If the first expression returns false or null, the ELSEIF clause evaluates another expression. An IF statement can have any number of ELSEIF clauses.

  Example of else-if:

  ```
  If (condition)
          Do this
  Else if(condition)
          Do this
  Else
          Do this
  ```

- **Nested-if:**
  When an if else statement is present inside the body of another "if" or "else" then this is called nested if else.

  Example of Nested if-else:

```
If (condition)
    Do this
Else{
    If(condition)
        Do this
    Else{
        Do this
        And this}}
```

Q1,b:

ANS:

```
#include <iostream>

using namespace std;

int main(){

    float num1, num2;

    cout<<"Please enter the first number: ";

    cin>> num1;

    cout<<"Please enter the second number: ";

    cin>> num2;


    if(num1>num2)

    cout<<"The first number '"<<num1<<"' is the LARGEST";

    else

    cout<<"The second number '"<<num2<<"' is the LARGEST";

return 0;}
```

Q2,a:

ANS:

Logical Operators:

A logical operator is a symbol or word used to connect two or more expressions such that the value of the compound expression produced depends only on that of the original expressions and on the meaning of the operator.

Common logical operators include AND, OR, and NOT.

AND Logical Operator: (&&)

- Used to combine two conditions
- true if both conditions are true

```
if ( gender == 1 && age >= 65 )
        senior++;
```

OR Logical Operator: (||)

- true if either of condition is true

```
if (semesterAvg >= 90 || finalExam >=90 )
  cout<<("Student grade is A“);
```

Not Logical Operator: (!)

- Returns true when its condition is false, & vice versa

```
if ( !( grade == 20 ) )
cout<<“hello world“);
```

Q2,b:

ANS:

```
#include <iostream>

using namespace std;

int main(){

        int F;

        cout<<"Please enter the temprature in fahrenheit: ";

        cin>> F;


        if(F>=40){

        cout<<"Very hot";}

        else if(F>=35 && F<40){

        cout<<"Tolerate";}

        else if(F>=30 && F<35){

        cout<<"Warm";}

        else if(F<30){

        cout<<"Cool";}

return 0;}
```

Q3,a:

ANS:

Loop:

In computer science, a loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things.

Types of Loops:

FOR loop:

A for loop enables a particular set of conditions to be executed repeatedly until a condition is satisfied.

Syntax of for loop:

```
for (initialization statement; test expression; update statement) {
 // statements
}
```

While loop:

A while loop is one of the most common types of loop. The main characteristic of a while loop is that it will repeat a set of instructions based on a condition. As far as the loop returns a boolean value of TRUE, the code inside it will keep repeating. We use this kind of loop when we don't know the exact number of times a code needs to be executed.

Syntax of while loop:

```
 while (condition) {
  //statements
  Initialization
}
```

Do-While loop:

If you recall the way the *for* and *while* loops work, you will remember that these loop types check for the loop condition at the beginning of the loop. Unless the condition is satisfied the loop will not be executed.

The do while loop checks the condition at the end of the loop. This means that the statements inside the loop body will be executed at least once even if the condition is never true.

Syntax of Do-while loop:

```
do {
 /* statement(s); */
 /*increment loop counter*/
} while ( condition );
```

Q3,b:

ANS:

```
#include<iostream>

using namespace std;

int main(){

    int num, R;

              cout << "Enter the number : ";

    cin >> num;

    R = num % 2;

    if (R == 0)

        cout << num << " is an even number. " << endl;

    else

        cout << num << " is an odd number. " << endl;

 return 0;}
```

Q4,a:

ANS:

The one-token statements continue and break may be used within loops to alter control flow.

Continue causes the next iteration of the loop to run immediately, whereas break terminates the loop and causes execution to resume after the loop.

Both control structures must appear in loops. Both break and continue scope to the most deeply nested loop, but pass through non-loop statements.

Although these control statements may seem undesirable because of their goto-like behavior, their judicious use can greatly improve readability by reducing the level of nesting or eliminating bookkeeping inside loops.

Q4,b:

ANS:

```
#include <iostream>

using namespace std;

int main(){

   int i,t=0;

   cout<<" The given numbers are: \n";

   for (i = 1; i <= 10; i++) {

        cout <<" "<< i << " ";

        t=t+i; }

   cout<<"\n The sum of all the given numbers = "<<t ;

return 0;}
```

Q5,a:

ANS:

Array:

- An array is a collection of data items, all of the same type, accessed using a common name.
- A one-dimensional array is like a list;
- A two dimensional array is like a table;
- The C language places no limits on the number of dimensions in an array, though specific implementations may.
- Some texts refer to one-dimensional arrays as vectors, and two-dimentional arrays as matrices, and use the general term arrays when the number of dimensions is unspecified or unimportant.

One-dimensional array:

A one-dimensional array is a type of linear array. Accessing its elements involves a single subscript which can either represent a row or column index.
As an example consider the C declaration int anArrayName[10]; which declares a one-dimensional array of ten integers.

Example:

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

int main() {
    int numbers[5] = {7, 5, 6, 12, 35};

    cout << "The numbers are: ";

    //  Printing array elements
    // using range based for loop
    for (const int &n : numbers) {
        cout << n << "  ";
    }


    cout << "\nThe numbers are: ";

    //  Printing array elements
    // using traditional for loop
    for (int i = 0; i < 5; ++i) {
        cout << numbers[i] << "  ";
    }

    return 0;
}
```

Two-dimensional array:

A 2D array has a type such as int[][] or String[][], with two pairs of square brackets. The elements of a 2D array are arranged in rows and columns, and the new operator for 2D arrays specifies both the number of rows and the number of columns.

Example:

```
#include<iostream.h>
#include<conio.h>

main(){
 int matrix [5] [5];
for (int m1=0 ; m1<5 ; m1++)
{
for (int m2=0 ; m2<5 ; m2++){
matrix [m1] [m2] = 5 ;
}}
getch();}
```

------------------------------------END---