Name	*Kaleem Ullah Khan*
ID	*7681*
Sec	*c*
Dep	*civil engineering*
Subject	*ICP (C++ Theory)*
Assignment	*Final term*
Semester	*8th*
Submitted to	*Engr Ashraf ali*

Q.1 (a): Write a program for yours Grading System using "If-else statement".

Ans;

```
#include<iostream>
using namespace std;
int main()
{ int marks;
cout<<" — -Program To Find Grade — -"<<endl;
cout<<"\nEnter Marks: ";
cin>>marks;
if(marks>=90 && marks<=100) cout<<"Your Grade is A .";
else if(marks>=80 && marks<90) cout<<"Your Grade is A.";
else if(marks>=70 && marks<80) cout<<"Your Grade is B.";
else if(marks>=60 && marks<70) cout<<"Your Grade is C.";
else if(marks>=50 && marks<60) cout<<"Your Grade is D.";
else if(marks>=0 && marks<50) cout<<"Your Grade is F.";
else cout<<"Invalid Marks."; return 0; }
```

Q No 1 (b)

Differentiate between "If statement" and "If-else statement".

Ans;

If statement;

The if statement is a decision-making structure that consists of an expression followed by one or more statements.

if statement in C/C++

if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if(condition)
{
   // Statements to execute if
   // condition is true
}
```

If-else statement

The if else is a decision-making structure in which the if statement can be followed by an optional else statement that executes when the expression is false.

if-else in C/C++

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won’t. But what if we want to do something else if the condition is false. Here comes the C else statement. We can use the else statement with if statement to execute a block of code when the condition is false.

Syntax:

```
if (condition)
{
   // Executes this block if
   // condition is true
}
else
{
   // Executes this block if
   // condition is false
}
```

Q.2 (a): Write a program to display a menu to perform various functions using "Switch statement".

Ans;

```
#include <iostream>

using namespace std;



int main(void)

{

char selection;



cout<<"\n Menu";

cout<<"\n========";

cout<<"\n A - Append";
```

```
cout<<"\n M - Modify";

cout<<"\n D - Delete";

cout<<"\n X - Exit";

cout<<"\n Enter selection: ";

// read the input

cin>>selection;



switch(selection)

{

case 'A' :

case 'a' :{cout<<"\n To append a record\n";}

break;
```

```
case 'M' :

case 'm' :{cout<<"\n To modify a record";}

break;

case 'D' :

case 'd' :{cout<<"\n To delete a record";}

break;

case 'X' :

case 'x' :{cout<<"\n To exit the menu";}

break;

// other than A, M, D and X...

default : cout<<"\n Invalid selection";
```

```
// no break in the default case

}

cout<<"\n";



return 0;

}
```

Q NO 2 (b)

Differentiate between “Nested If-else statement” and “Switch statement”.

Ans;

switch vs if else

Prerequisite – Switch Statement, Decision making(if else)

A switch statement is usually more efficient than a set of nested ifs. Deciding whether to use if-then-else statements or a switch statement is based on readability and the expression that the statement is testing.

Check the Testing Expression: An if-then-else statement can test expressions based on ranges of values or conditions, whereas a switch statement tests expressions based only on a single integer, enumerated value, or String object.

Switch better for Multi way branching: When compiler compiles a switch statement, it will inspect each of the case constants and create a “jump table” that it will use for selecting the path of execution depending on the value of the expression. Therefore, if we need to select among a large group of values, a switch statement will run much faster than the equivalent logic coded using a sequence of if-elses. The compiler can do this because it knows that the case constants are all the same type and simply must be compared for equality with the switch expression, while in case of if expressions, the compiler has no such knowledge.

if-else better for boolean values: If-else conditional branches are great for variable conditions that result into a boolean, whereas switch statements are great for fixed data values.

Speed: A switch statement might prove to be faster than ifs provided number of cases are good. If there are only few cases, it might not effect the speed in any case. Prefer switch if the number of cases are more than 5 otherwise, you may use if-else too

Q.3 (a): Differentiate between “Relational operator” and “Relational Expression”.

Ans;

Relational operator

A relational operator is a programming language construct or operator that tests or defines some kind of relation between two entities. These include numerical equality (e.g., 5 = 5) and inequalities (e.g., 4 ≥ 3).[1]

Discussion

The relational operators are often used to create a test expression that controls program flow. This type of expression is also known as a Boolean expression because they create a Boolean answer or value when evaluated. There are six common relational operators that give a Boolean value by comparing (showing the relationship) between two operands. If the operands are of different data types, implicit promotion occurs to convert the operands to the same data type

Relational Expression

A relational expression consists of two arithmetic expressions or two character expressions separated by a relational operator. A relational operator tests for a relationship between the two expressions. The value of the relational expression is either .TRUE. or .FALSE. depending on whether the stated relationship holds.

Fortran supports the following relational operators:

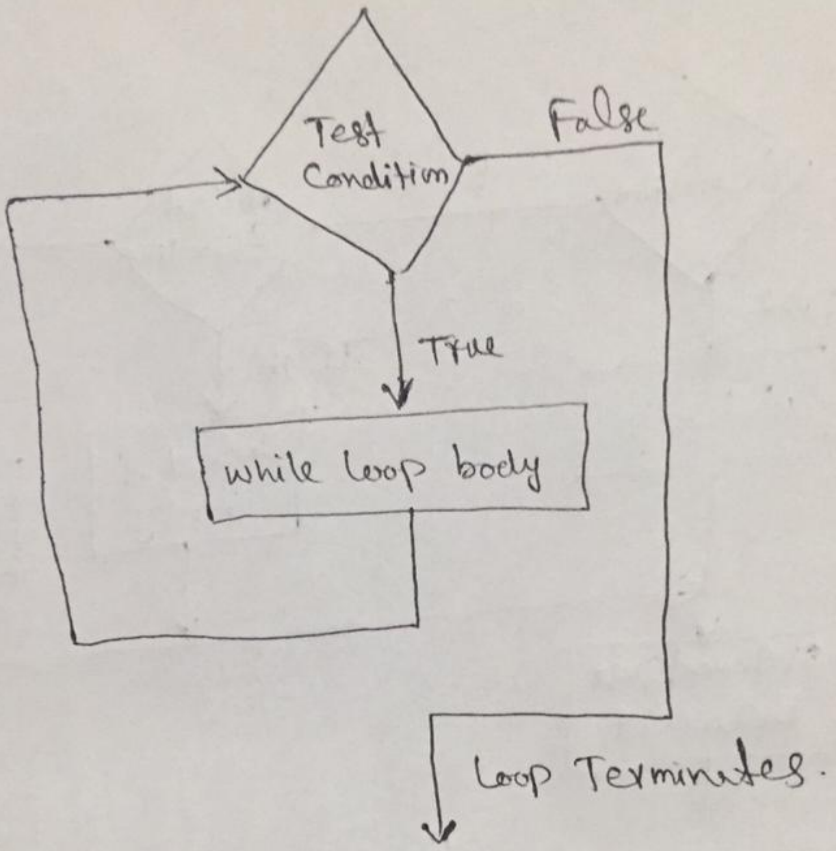
Operator	Relationship
.LT.	Less than
.LE.	Less than or equal to
.EQ.	Equal to
.NE.	Not equal to
.GT.	Greater than
.GE.	Greater than or equal to

Both delimiting periods are required.

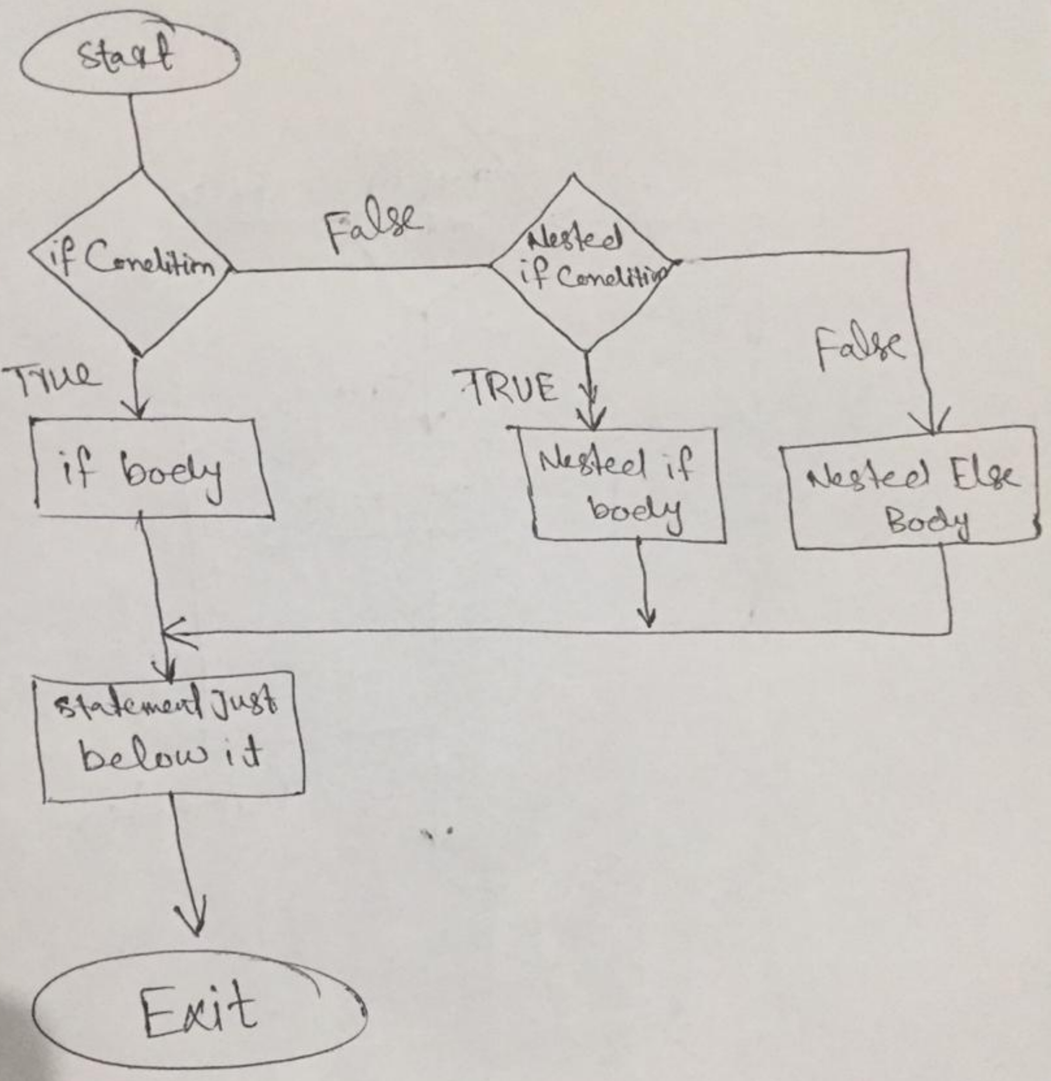
Complex expressions can be related only by the .EQ. and .NE. operators. Complex entities are equal if their corresponding real and imaginary parts are both equal.

In an arithmetic relational expression, the arithmetic expressions are first evaluated to obtain their values. These values are then compared to determine whether the relationship stated by the operator is valid

Q3 Flow chart of while Loop:
(b)



Q3
(b) Nested-if-else Flow diagram:



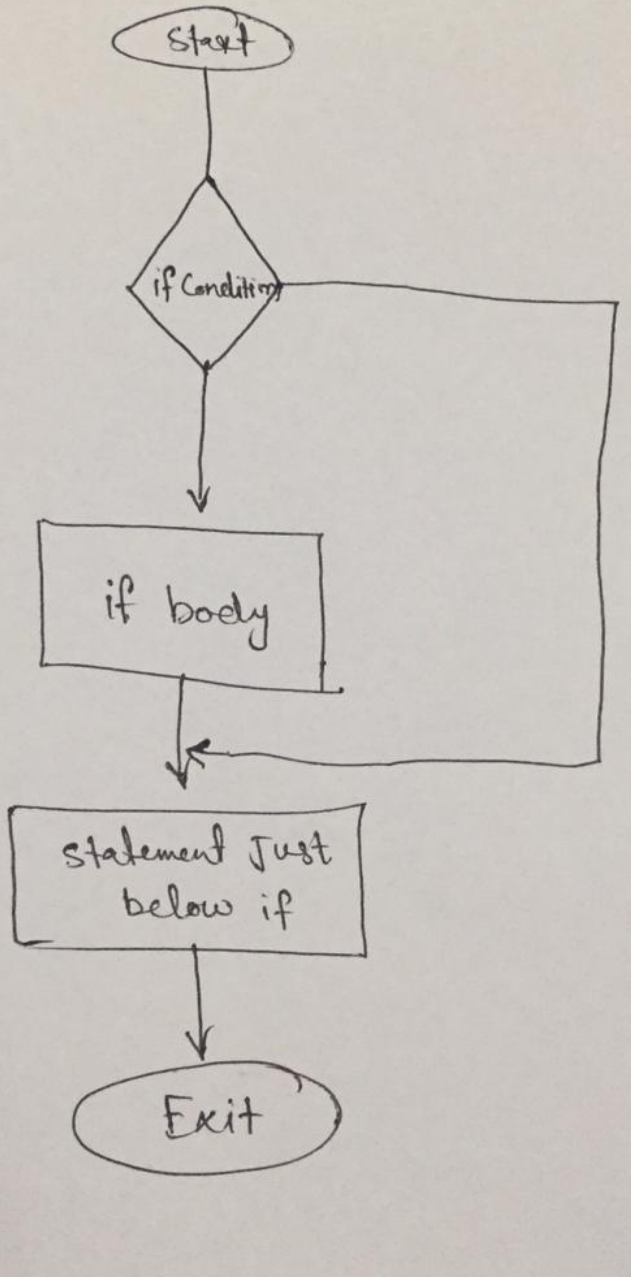
Q.4 (a): Write a program in C++ to find the Volume of a cylinder.

Ans;

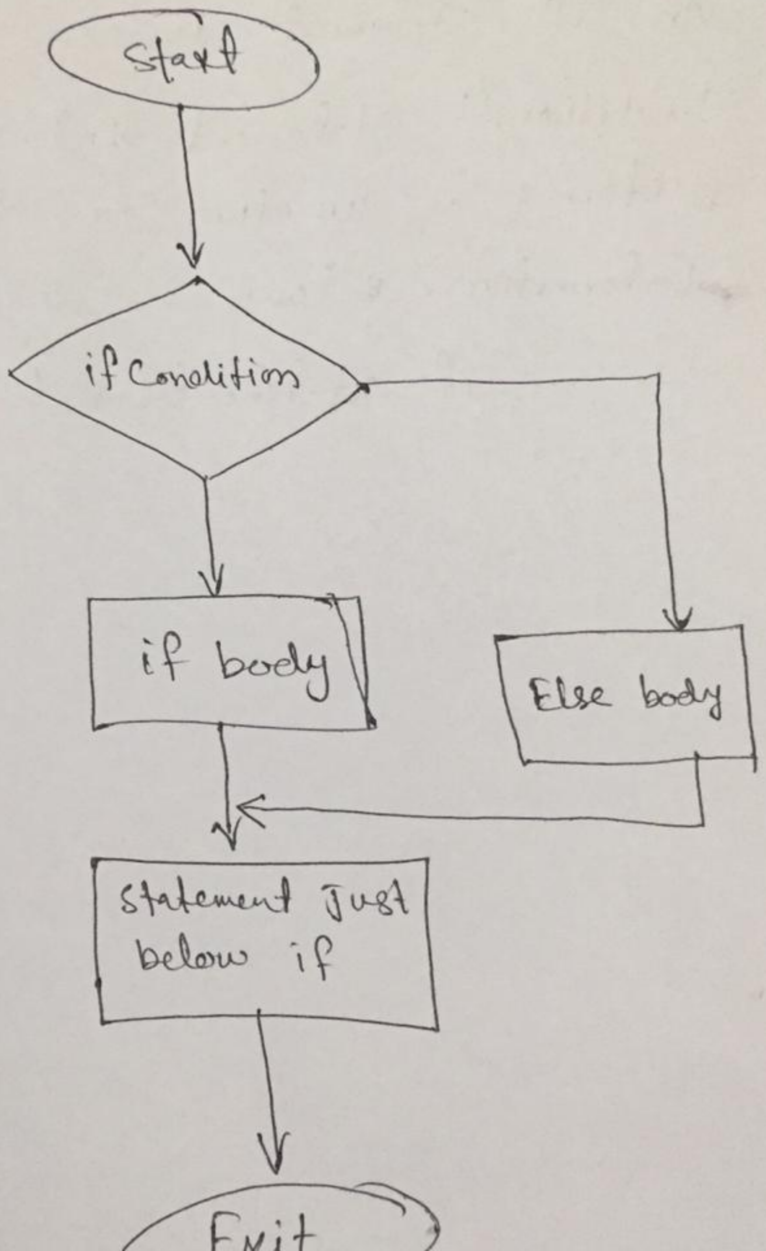
```
#include <iostream>
using namespace std;


   int main()
   {
    int rad1,hgt;
    float volcy;
            cout << "\n\n Calculate the volume of a cylinder :\n";
            cout << "----------------------------------------\n";
       cout<<" Input the radius of the cylinder : ";
    cin>>rad1;
            cout<<" Input the height of the cylinder : ";
    cin>>hgt;
    volcy=(3.14*rad1*rad1*hgt);
       cout<<" The volume of a cylinder is : "<< volcy << endl;
       cout << endl;
       return 0;
    }
```

Q4
(b) if statement flow diagram:



if-else flow diagram:



Q.5 (a): What is Sequential Statement?

Ans;

Sequential statements like A := 3 are interpreted one after another, in the order in which they are written. VHDL sequential statements can appear only in a process or subprogram.

A VHDL process is a group of sequential statements; a subprogram is a procedure or function.

To familiarize yourself with sequential statements, consider the following:

Assignment Statements

Variable Assignment Statement

Signal Assignment Statement

if Statement

case Statement

loop Statements

next Statement

exit Statement

Subprograms

return Statement

wait Statement

null Statement

Processes are composed of sequential statements, but processes are themselves concurrent statements .

All processes in a design execute concurrently. Howev-
er, at any given time only one sequential statement is
interpreted within each process.
A process communicates with the rest of a design by reading
or writing values to and from signals or ports declared outside
the process.
Sequential algorithms can be expressed as subprograms and
called sequentially (as described in this chapter) or concur-
rently .
Sequential statements are
assignment statements
that assign values to variables and signals.
flow control statements
that conditionally execute statements (if and case),
repeat statements (for...loop), and skip statements
(next and exit).
subprograms
that define sequential algorithms for repeated use in a
design (procedure and function).
wait statement
to pause until an event occurs (wait)
null statement
to note that no action is necessary (null).

Q5(b): Write a program which performs the arithmetic operation by using all arithmetic

operators.

Ans;

```
#include<iostream.h>
#include<conio.h>
void main()
{
        int a,b,c,d,e,f,g;
        clrscr();
        cout<<"\n Enter First Number a : ";
        cin>>a;
        cout<<"\n Enter Second Number b : ";
        cin>>b;
        c=a+b;
        d=a-b;
        e=a*b;
        f=a/b;
        g=a%b;
        cout<<" Addition = "<<c<<"\n";
        cout<<" Subtraction = "<<d<<"\n";
        cout<<" Multiplication = "<<e<<"\n";
        cout<<" Division = "<<f<<"\n";
        cout<<" Modulus = "<<g<<"\n";
        getch();
}
```