Unit-4 Decision Statements

1. What is Conditional Statement? Give list of Conditional Statements.

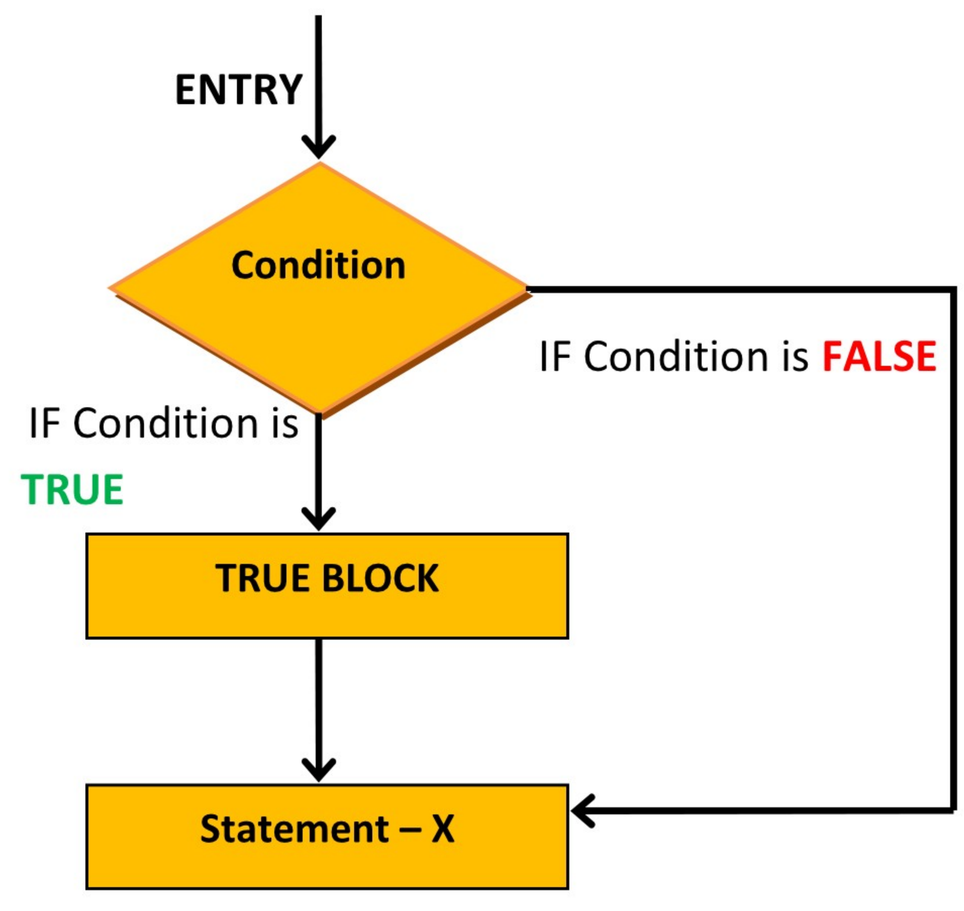
- Conditional statements help you to make a decision based on certain conditions. These conditions are specified by a set of conditional statements having Boolean expressions which are evaluated to a Boolean value true or false. There are following types of conditional statements in C.
  1. If Statement
  2. If-Else Statement
  3. Nested If Statement
  4. If-Else-If Ladder
  5. Switch Statement

2. Explain if condition statement with example and draw flowchart.

- If is used to control the flow of execution of statements.
- It is two way decision making statements.
- It evaluates expression first and based on its result, the control is transferred to the particular statement.
- The general form of simple *if* statement is

```
if (condition)
{
True block
}
statement-x;
```

- If the condition is true, the true block will be executed; otherwise the true block will be skipped and the execution will jump to the statement-x.



Example:

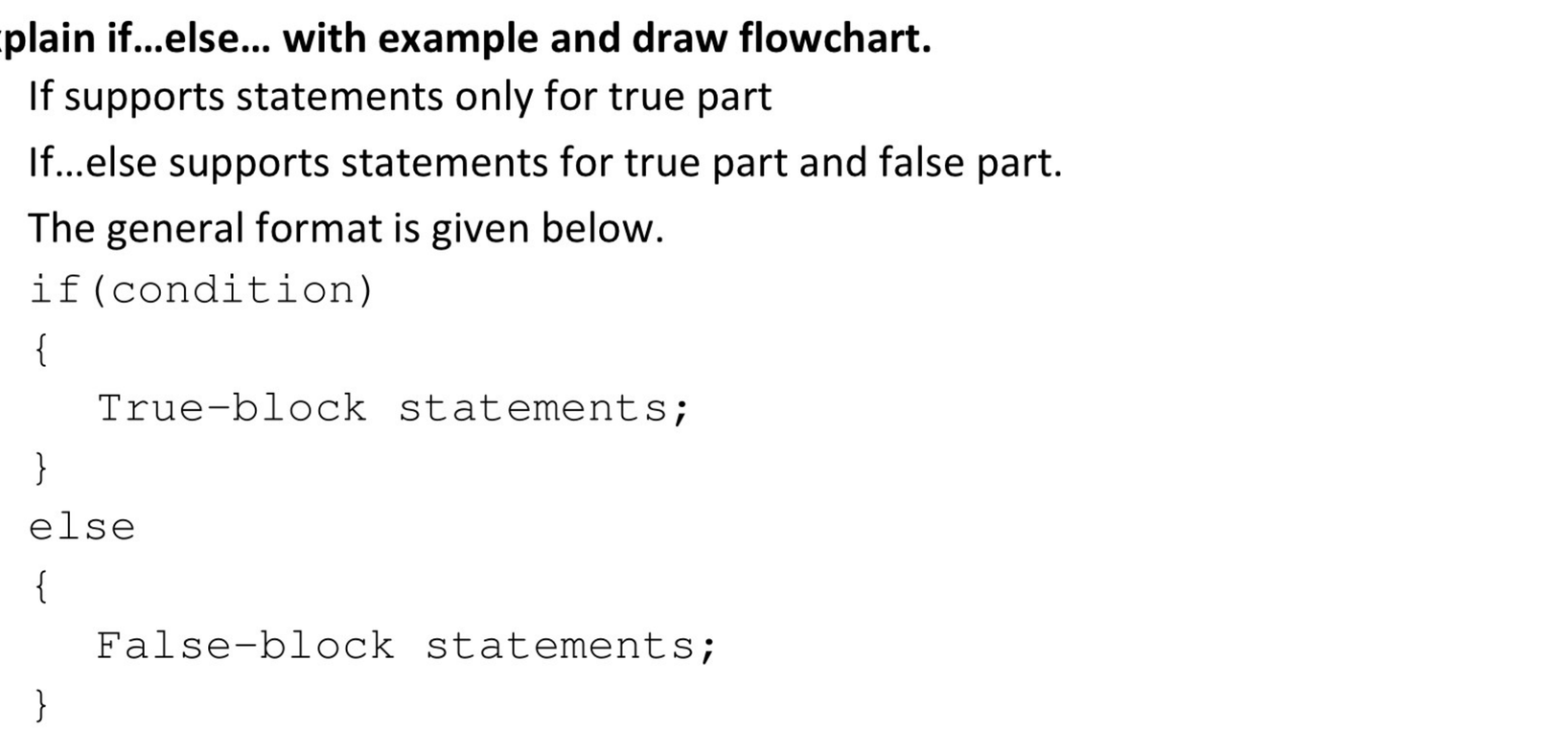
```
#include<stdio.h>
void main()
{
     int no;
 printf("Enter the number:");
 scanf("%d", &no);
 if(no%2==0)
{
      printf("no is even");
}
 if(no%2!=0)
{
      printf("no is odd");
}
}
```

3. Explain if…else… with example and draw flowchart.

- If supports statements only for true part
- If…else supports statements for true part and false part.
- The general format is given below.

```
if(condition)
{
   True-block statements;
}
else
{
   False-block statements;
}
statement-x;
```

- If condition is true, then true-block statement is executed otherwise false-block statement is executed.
- Every time, either true block or false block will be executed.
- In both the cases, statement-x will be executed immediately after that block.



Example: Check whether the given number is even or odd.

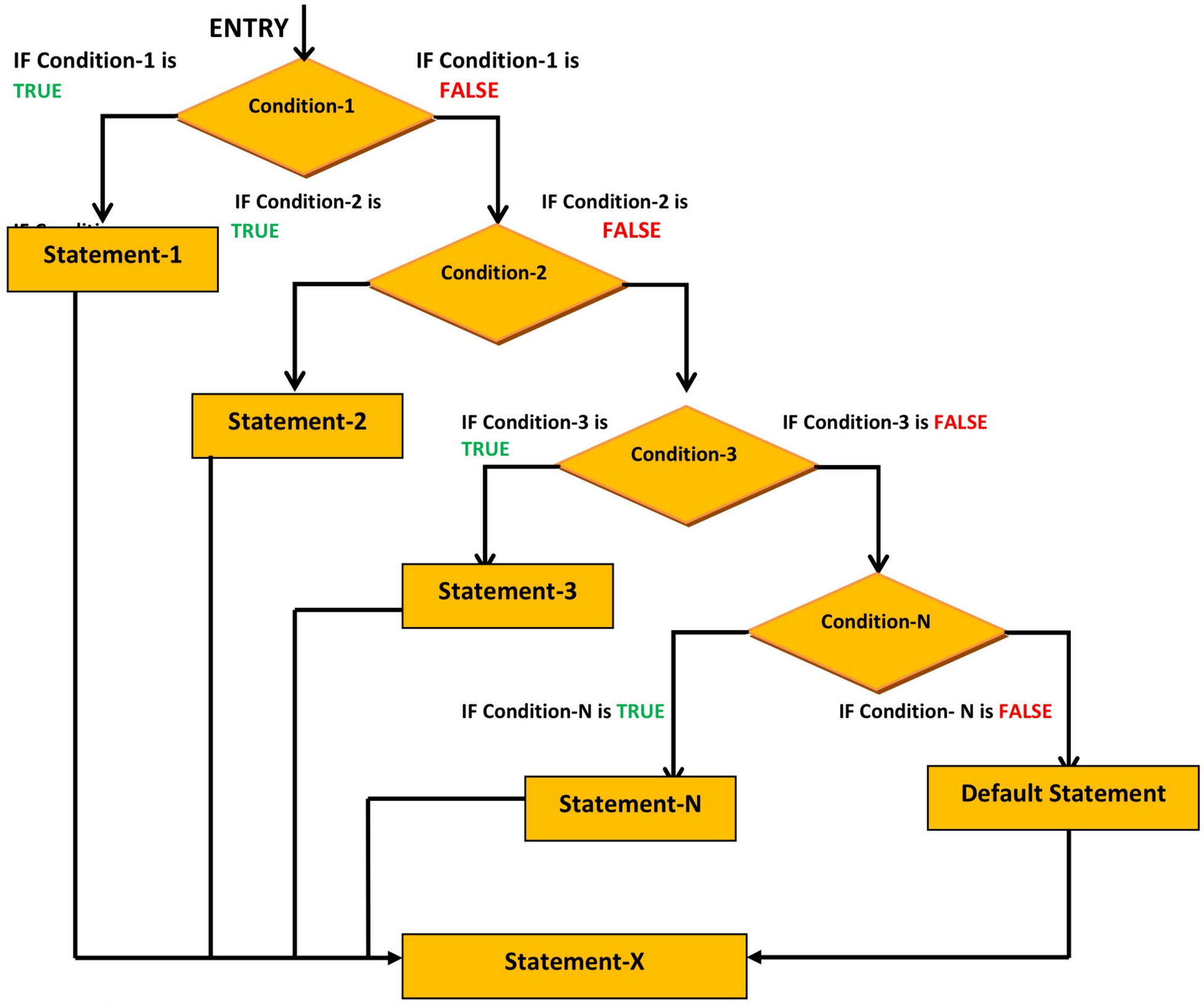
```
#include<stdio.h>
void main()
{
      int no;
      printf("Enter the number:");
      scanf("%d", &no);
      if(no%2==0)
      {
            printf("the number is even");
      }
      else
      {
            printf("the number is odd");
      }
}
```

4. Explain if…else...if ladder with example and draw flowchart.

- When multipath decisions are involved, we may use if…else…if ladder, which takes the following general form.
- The general format is given below.

```
if(condition-1)
{
   statement-1;
}
else if(condition-2)
{
   statement-2;
}
else if(condition-N)
{
   statement-N;
}
else
{
   default-statement;
}
statement-x;
```

- First condition-1 is checked and if it is true, then statement-1 will be executed and control goes to statement-x.
- If condition-1 is false, then condition-2 is checked and if it is true then statement-2 will be executed and control goes to statement-x.
- If condition-2 is false, then condition-3 is checked and process repeats
- This process is repeated until either it finds one of the conditions is true or all the conditions are false. If all the conditions are false, then default-statement will be executed.
- Else part is optional in if…else…if ladder.




```
#include<stdio.h>
void main()
{
      int a,b,c;
      printf("Enter a,b,c:");
      scanf("%d%d%d", &a, &b, &c);
      if(a>b && a>c)
      {
           printf(“a is maximum”);
      }
      else if(b>a && b>c)
      {
           printf("b is maximum");
      }
      else
      {
           printf("c is maximum");
      }
}
```

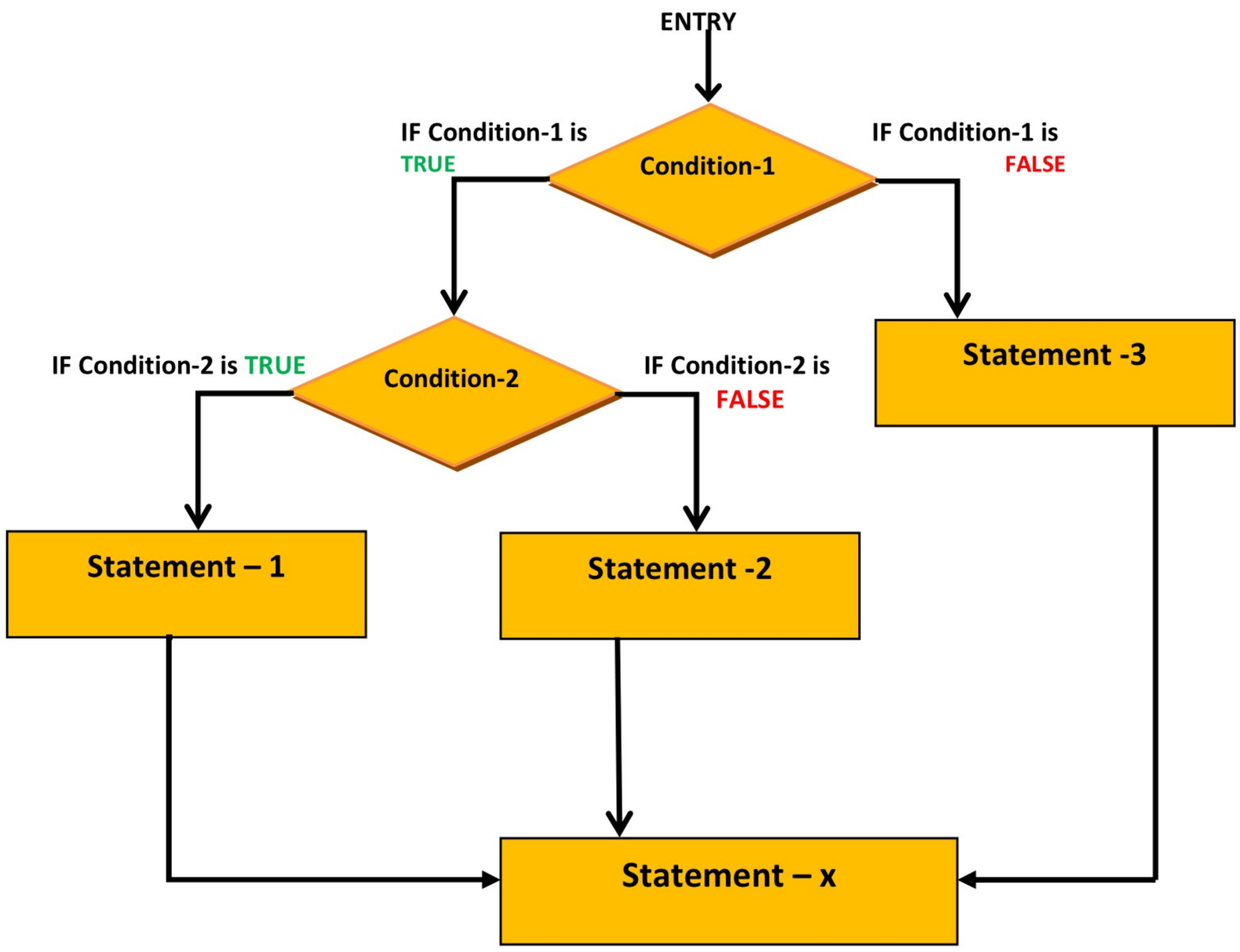
5. Explain nested if with example and draw flowchart.

- When multipath decisions are involved, we may use *if…else…if* ladder, which takes the following general form.

```
if(condition-1)
{
        if(condition-2)

        {
            statement-1;
        }
        else
        {
            statement-2;
        }
}
else
{
     statement-3;
}
statement-x;
```

- If condition-1 is true then condition-2 is evaluated. If it is true then Statement-1 will be executed, if it is false then Statement-2 will be executed.
- If condition-1 is false then Statement-3 will be executed.



```
#include<stdio.h>
void main()
{
      int a,b,c;
      printf("Enter value of a, b, c:");
      scanf("%d%d%d", &a, &b, &c);
      if(a>b)
      {
            if(a>c)
            {
                  printf("a is max");
            }
            else
            {
                  printf("c is max");
            }
      }
      else
      {
            if(b>c)
            {
                  printf("b is max");
            }

            else
            {
                  printf("c is max");
            }
      }
}
```

6. Explain switch-case statement with example.

- The switch statement is a multi-way decision making.
- It tests whether an expression matches any one of the constant values or not.
- General form of switch-case is as below.

```
switch(expression){
      case const-expr 1:
      statement 1;
      break;
      case const-expr 2:
      statement 2;
      break;
      case const-expr 3:
      statement 3;
      break;
      default:
      statements;
      }
```

- Expression in switch should be integer or character expression. Float or any other data type is not allowed.
- Each case is labeled by one or more integer-valued constant.
- If a case matches the expression value then execution starts at that case.
- Value of all case expressions must be different.
- If none of the cases are matched then default case is executed. default case is optional.
- The break statement causes an immediate exit from the switch.

Example:

```
#include<stdio.h>
void main()
{
    int grade;
printf(“Enter Your Choice: ”);
scanf(“%d”,&grade);
switch (grade);
{
 case 1:
 printf("Fall (F)\n");
 break;
 case 2:
 printf("Bad (D)\n");
 break;
 case 3:
 printf("Good (C)\n");
 break;
 case 4:
 printf("Very Good (B)\n");
 break;
 case 5:
 printf("Excellent (A)\n");
 break;
 default:
 printf("You have inputted false grade\n");
 break;  // break isn’t necessary here
}
}
```

Rules for switch statement:

- The switch expression must be integral type. Float or other data types are not allowed.
- Case labels must be constant or constant expression. And They must be unique.
- Case labels must end with colons.
- The break statement is optional.
- The default case statement is optional.
- Nesting of switch statement is allowed.

7. State difference between if-else and switch-case.

if-else	switch
IF Statement is used to select either of one statement.	Switch statement is used to select one statement from more than two statement.
If statement's value may be based on constraints.	Switch statement's value base on user's choice.
If statement's check line by line value.	Switch search by Binary.
Float, double, char, int and other are used in If statement.	Only int and char datatype are use in switch statement. It is use to select either one of statement from two statement.

8. Explain the break, goto and Continue Statement with example.

- Sometimes we need to terminate the loop when some conditions went wrong.
- Suppose if we need to find a number among 100 numbers set and when we find that number the loop should have to be terminated.

Break Statement:

- Break statement is used when we need to exit from the loop.
- Break statement provides us the facility to exit the loop in the for, while, do…..while and switch looping statements.

```
#include<stdio.h>
void  main()
     {
          int i=0, sum=0;
          float sum=0, average;
          printf ("input the marks, -1 to end\n");
          while (I >=0)
          {
               scanf ("%d", & I);
               if (I==-1)
               {
                  break;
               }
               sum+= I;
          }
          printf ("%d", sum);
}
```

Continue Statement:

- Continue statement will skip the remaining part of the body of the iterative loop.
- Continue statement will inform the complier to skip the remaining part of the body and continue the next iteration.

```
#include<stdio.h>
void  main()
   {
```

```
        int   i=1 num,sum=0;
        for (i=0; i<5; i++ )
        {
             scanf ("%d", &num);
             if (num < 0)
             {
             continue;  //starts with the beginning of loop
             } sum + = num;
        }
printf("the sum of positive numbers entered = %d", sum);
}
```

Goto Statement:

- Goto statement provides us the facility to jump from one point to another point.
- Goto statement should have to be labeled.
- Lable statement can be used anywhere in the function.
- The goto statement is not used efficiently because it is tough to debug the goto statement and it is time consuming as well.

  Example :- following program prints 10,9,8,7,6,5,4,3,2,1

```
#include<stdio.h>
   void main ()
        {
             int n=10;
             loop:
             printf("%d,",n);
             n--;
             if (n>0)
                  goto loop;
        }
```

9. Why goto statement is avoided?

- The use of GoTo statement may lead to code that is buggy and hard to follow.
- Because of GoTo you may lead to jump out of scope.
- It makes difficult to trace the flow for the programmer.
- After applying GoTo, it becames hard to modify the program.