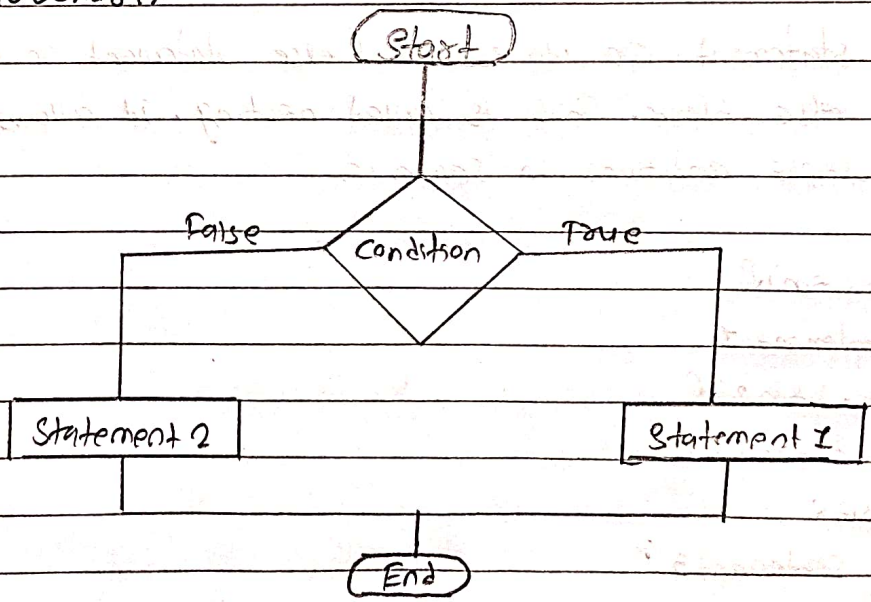
1) Explain the working of if-else and nested if-else with a flow chart. provide an example to determine if a number is positive, negative, or zero.

→ The if-else statement allows the program to execute one block of code if the condition is true and another block of code if the condition is false.

Syntax:
```
if (condition)
{ statement1; }
else
{ statement2; }
```

Flowchart:



Example: to check a person can vote or not

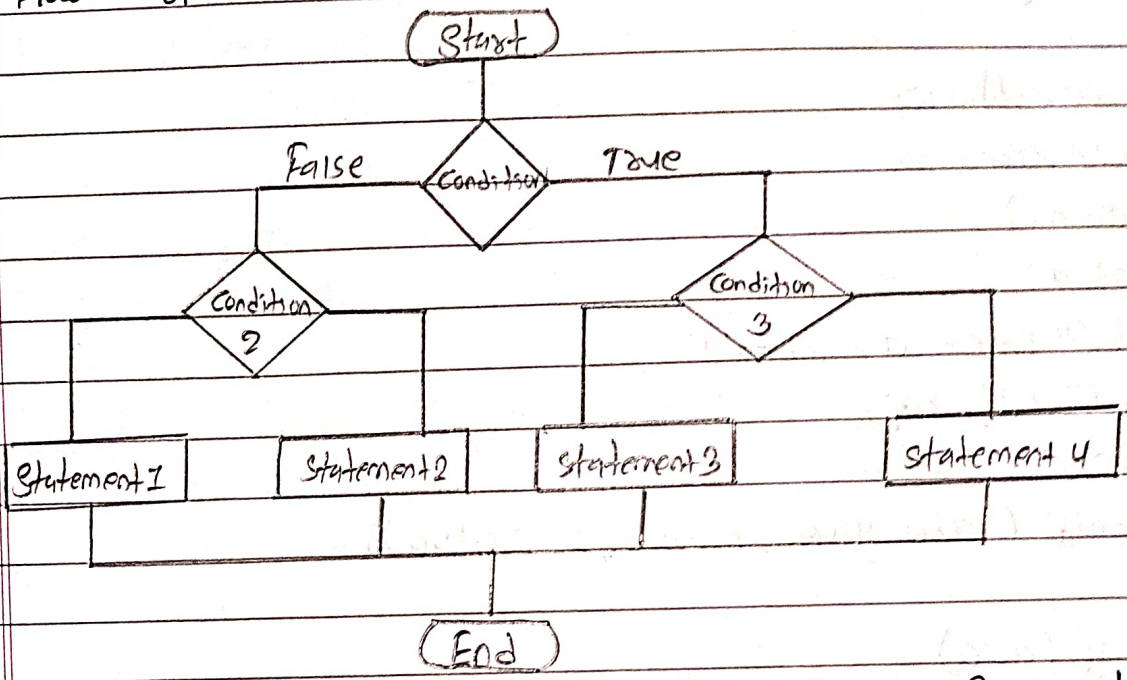
```
#include <stdio.h>
#include <conio.h>
int main()
{ int a;
  printf("Enter your age\t");
  scanf("%d", &a);
if (a>=18) {
    printf("You are eligible for voting"); }
else{
    printf("You are not eligible for voting");}
getch(); return 0;
}
```

→ Nested-if-else statement can place one if else statement inside another if or else block. This is called nesting. It allows us to check multiple conditions in sequence.

Syntax:

```
if (condition){
    statement 1
if (condition 2){
    statement 2 }
else{
    statement 3}
}
else {
    statement 4}
}
```

Flow-chart



Example: Distinction, first & second division of marks of student.

```
#include<stdio.h>
#include<conio.h>
int main(){
  int a;
printf("Enter your marks");
scanf("%d",&a);
if (a>=75) {
  printf(" Distinction"); }
else {
   if (marks>=60){
    printf(" first division"); }
  else{
     printf (" second division"); }
  }
getch(); return 0;
  }
}
```

Programming:

```
#include<stdio.h>
#include<conio.h>
 int main()
{ int a;
printf("Enter a number");
scanf("%d", &a);
if (a>0)
{ printf ("The given number is positive");
}
else if (a<0)
{
  printf ("The given number is negative");
}
else
{
printf ("The given number is zero");
}
getch();
return 0;
}
```

② Describe the for loop with multiple initializations. Draw a flowchart and provide an example to print the first 10 odd numbers.

→ The for loop is used when the number of repetation is known, it involves initialization, condition and increment, decrement.

Syntax with multiple initialization:

```
for (initialization 1, initialization 2; condition; update 1, update 2)
 { statement }
```

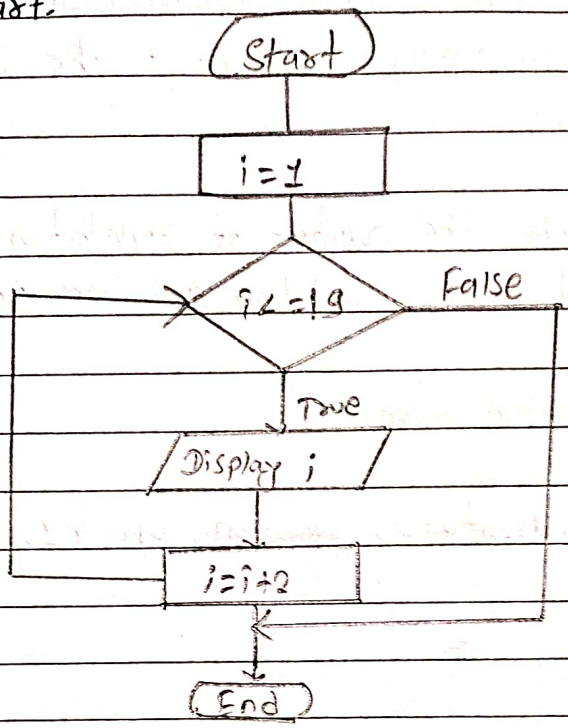
Example:

```
#include <stdio.h>
#include <conio.h>
int main()
{
 for (int i=1, j=10; i<=5 && j>=6; i++, j--)
{
 printf(" i = %d, j = %d \n", i, j);
}
getch();
return 0;
}
```

output:

i = 1, j = 10
i = 2, j = 9
i = 3, j = 8
i = 4, j = 7
i = 5, j = 6

Flow chart:



Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
  int i;
for(i=1 ; i<=19 ; i+=2)
{
 printf("%d\n",i);
} getch();
return 0; }
```

③ Explain how the while loop works with break and continue statements. Draw a flowchart and provide an example to skip multiples of 3 and stop the loop when the number reaches 8.

→ The while loop in c repeatedly executes a block of code as long as the given condition is true.

* Break
- The break statement is used to immediately exit the loop, regardless of the loop's condition.
- once break is executed, the program moves to the code after the loop

Example:

```
#include<stdio.h>
#include<conio.h>
int main() {
  int i=1;
while (i<=10)
  {
   if (i==5)
   {
     break;
   }
printf("%d\n",i);
 i++;
} getch();
  return 0;
}
```

output:

1
2
3
4

Continue

- The continue statement skip the current execution of the loop and moves to the next execution.
- It doesn't exit the loop but skips everything after the continue for the execution.

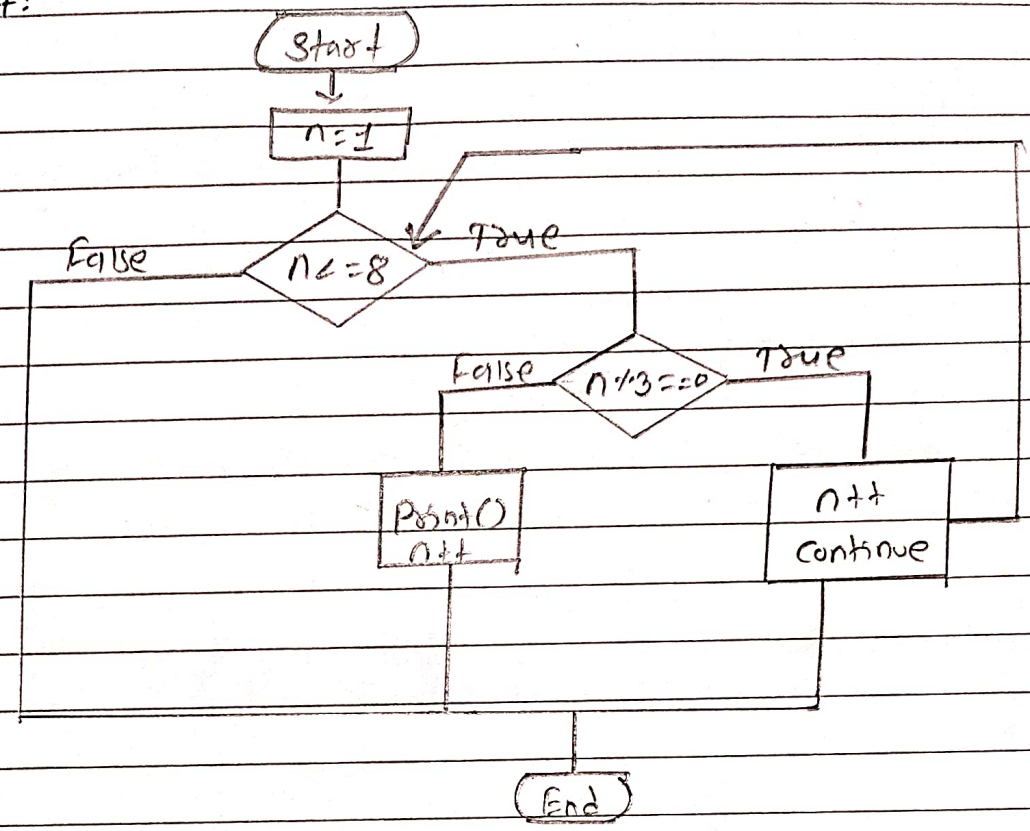
Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{ int i=0;
while (i<10)
 {
  i++;
 if (i==5)
 {
  continue;
 }
 printf("%d\n",i);
}
getch();
return 0;
}
```

output:

1
2
3
4
6
7
8
9
10

Flow chart:



Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
  int n=1;
while (n<=8){
    if (n%3==0)
    { n++;
      continue;
    }
printf("%d\n",n);
  n++;
}getch(); return 0;
}
```

④ Compare the switch statement and if-else ladder with flow charts.

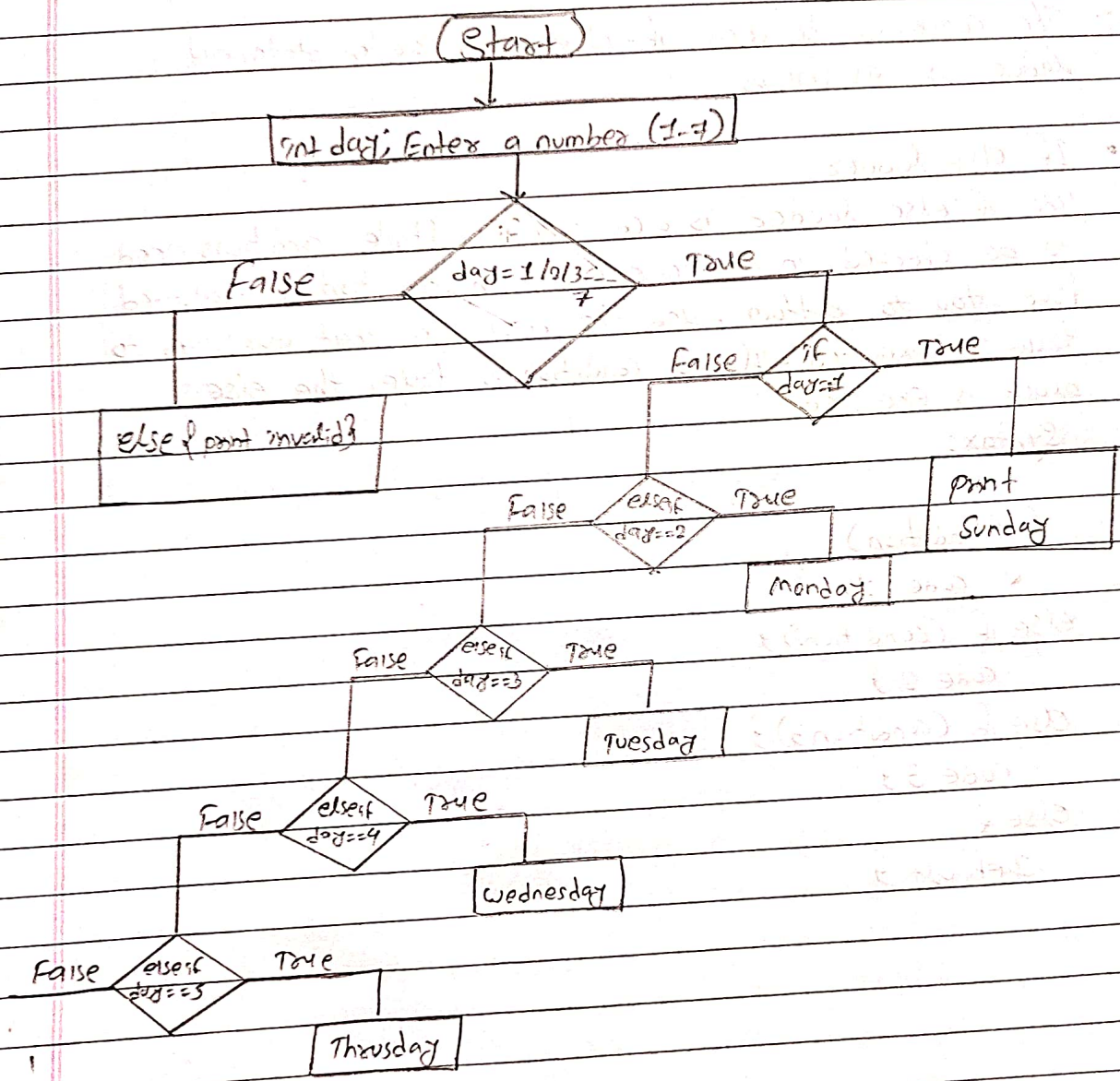
→ The comparison between if-else and switch statement ladder is as follows:-

- If-else ladder:
The if else ladder is used when multiple conditions need to be checked in a sequence. Each condition is evaluated from top to bottom, and the first one that evaluates to true is executed. If no condition is true, the else block is executed.

Syntax:

```
if (condition)
  { code 1 }
else if (condition2) {
    code 2 }
else if (condition3) {
    code 3 }
else {
  Default }
```

Flow-chart: Here we will enter 1 to 7 any one to print respective day of week.



This will continue till it doesn't find true value i.e day must be from 1 to 7.

Similar program:

```
#include<stdio.h>
#include<conio.h>
int main(){
 int day;
 printf("Enter a number (1-7) for the day of the week:");
 scanf("%d", &day);
 if (day==1){
  printf("Sunday\n"); }
else if (day==2){
  printf("Monday\n"); }
else if (day==3){
 printf("Tuesday\n"); }
else if (day==4){
 printf("Wednesday\n"); }
else if (day==5){
 printf("Thursday\n"); }
else if (day==6){
 printf("Friday\n"); }
else if (day==7){
 printf("Saturday\n"); }
else
{ printf("Invalid input! plese enter a valid number\n");
}
getch();
return 0;
}
```

- Switch Statement

- The switch statement tests a single expression (variable) against multiple possible values. Each value (case) is checked, and when a match is found, the corresponding block is executed. If no match is found, the default block is executed.

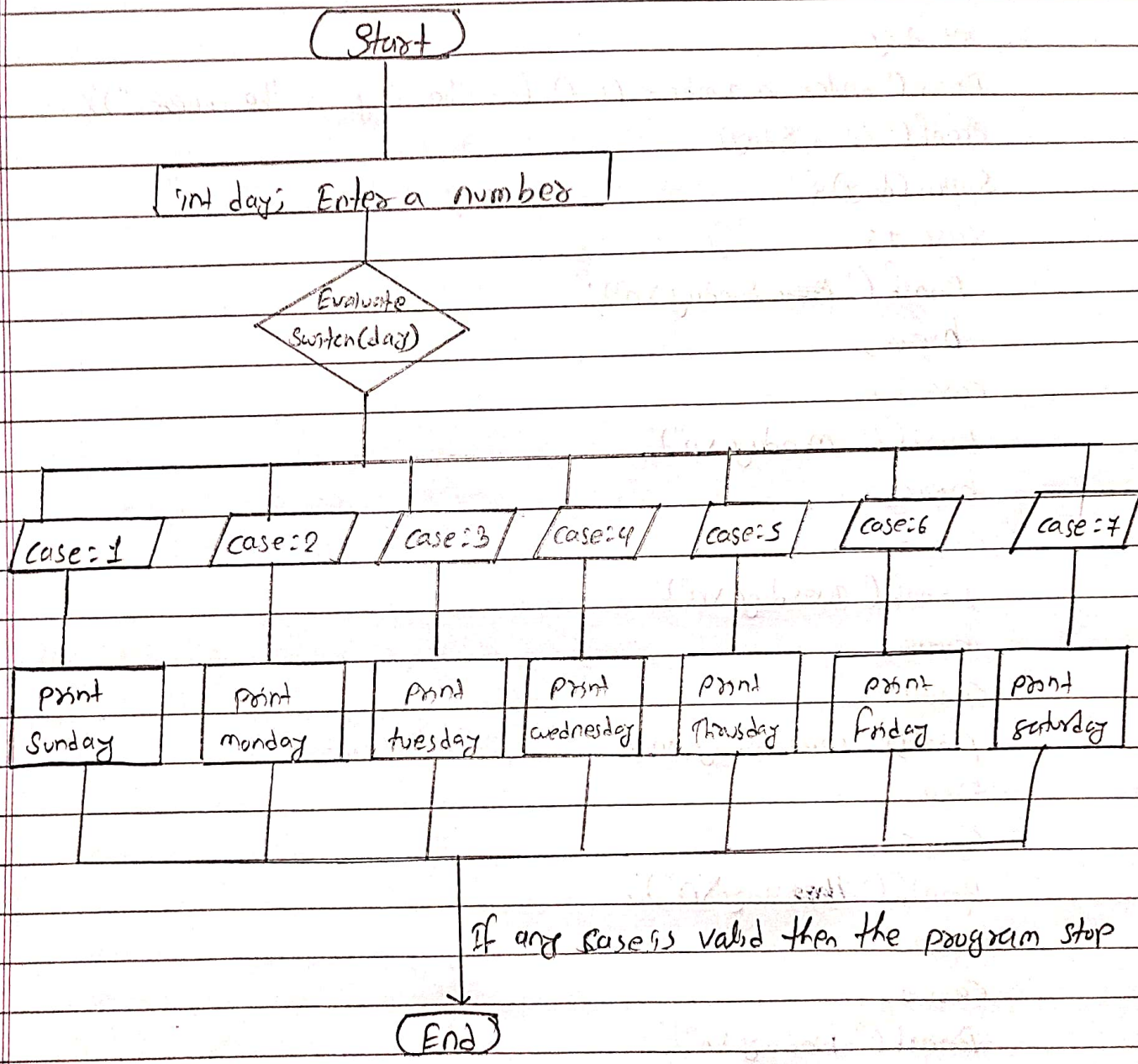
Syntax:

```
Switch (expression)
{ case value1;
   break;
  case value 2;
   break;
  case value 3;
   break;
  default;
}
```

Note:

- It works only with discrete values like integers and characters
- Handled with the default block
- Break is needed to prevent

Flowchart : Same as previous, enter a number 1-7 to print respective days of week.



```
#include<stdio.h>
#include<conio.h>
int main(){
 int day;
 printf("Enter a number (1-7) for the day of the week:");
 scanf("%d", &day);
 switch(day){
  case 1;
  printf("Sunday\n");
  break;
 case 2;
 printf("monday\n");
 break;
 case 3;
 printf("tuesday\n");
 break;
 case4;
 printf("wednesday\n");
 break;
 case5;
 printf("Thursday\n");
break;
 case6;
 printf("friday\n");
 break;
 case 7; printf("saturday"); break;
 default:
  printf("Invalid input! Plese enter valid number"); }
 getch(); return 0;
 }
```

⑤ Explain the working of the do-while loop. Draw a flowchart and provide an example where the loop continues asking for a new number until user enter zero.

→ The do-while loop is similar to the while loop but the condition is checked after the code block executes. This means that the code inside the loop will execute at least once, even if the condition is false.

Syntax:

```
do{
    statement;
}
while (condition);
```

Example:

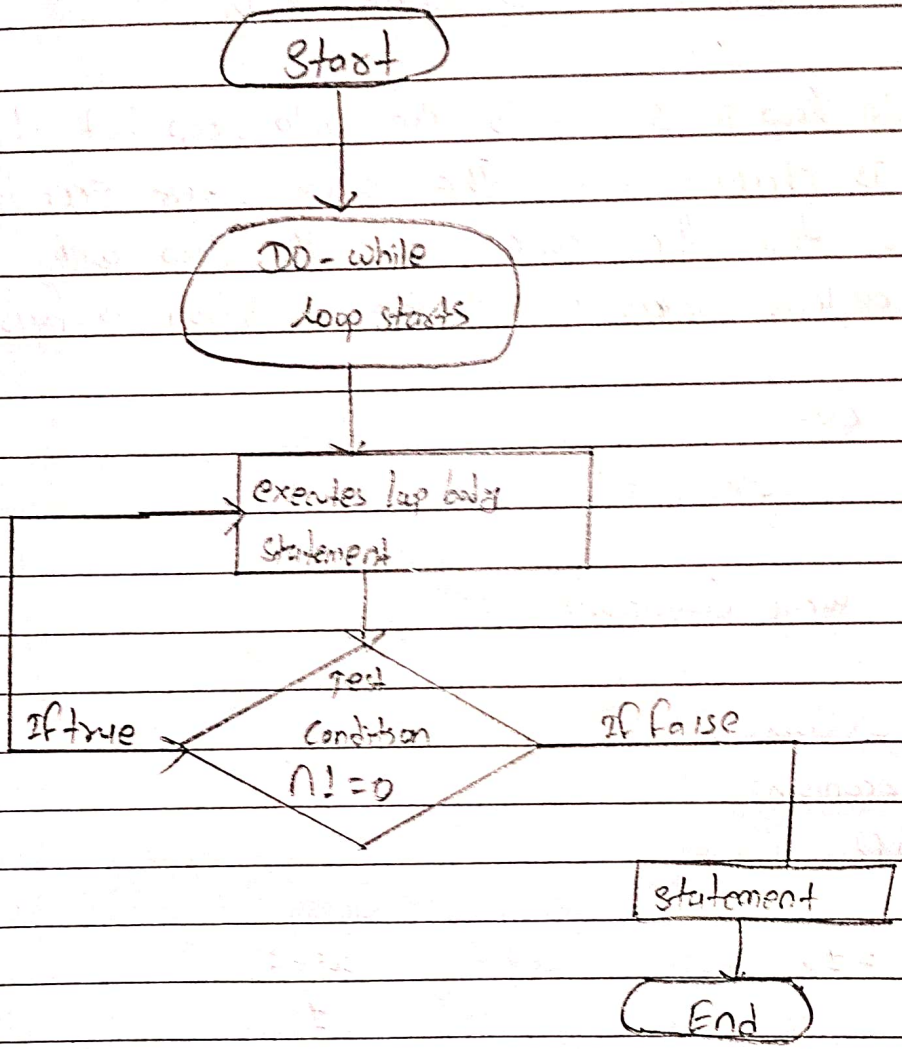
```
#include <stdio.h>
#include <conio.h>
int main()
{
  int n = 1;
  do {
     printf("%d \n", n);
     n++;
  } while (n<=5);
  getch() return 0;
}
```

output:

1
2
3
4
5

- The do-while loop is useful when you need to ensure that a block of code runs at least once
- It consists of a block followed by a while condition.

Flowchart:


Start
DO - while
loop starts
executes loop body
statement
test
condition
n! = 0
If true
If false
statement
End

program:

```
#include <stdio.h>
#include <conio.h>
int main()
 {
  int n;
  do
   {
    printf("Enter a number (0 to exit loop): ");
    scanf("%d", &n);
  }
  while(n!=0);
printf("You exited the loop successfully \n");
getch();
return 0;
}
```