Module 2

Control Statements in C

CONTROL STATEMENTS IN C

Ques 1) What is control statement? List out the different types of control statements available in C.

Ans: Control Statements

C provides decision making statements also known as **control statements**. Control flow is very sensitive part of the program. The execution sequence of the program statements is handled by the control flow. There are two types of control structure in C:

1) Selection or Branching:
   i) Conditional Branching Statements:
      a) if
      b) if-else
      c) if-else-if
      d) switch
   ii) Unconditional Branching Statements/Jump Statements:
      a) return
      b) break
      c) continue
      d) goto
2) Iteration or Loop:
   i) for
   ii) while
   iii) do-while

Ques 2) Explain the 'if' and 'if-else' statement.

Or

Write the syntax and also draw a flowchart for 'if' and 'if-else' statement.

Ans: if Statement

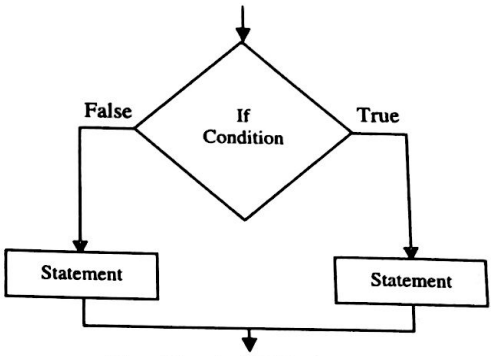
If the Boolean expression evaluates to true, then the 'if' block will be executed.

Syntax:

```
if (expression)
{
        statement;
}
```

The statements included are single, block of statements or no statements (i.e., empty statements) and else is optional statement.

The flowchart for 'if' statement is shown below:



Flow Chart of if Statement

For example, let consider the following code:

```
if ( 5 < 10 )
    printf( "Five is now less than ten" );
```

Here, we are just evaluating the statement, "5 < 10", to see if it is true or not. This statement will print "Five is now less than ten" because 5 is always less than 10.

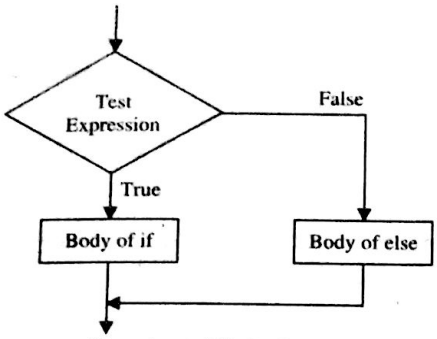
if-else Statement

In case the value of if expression is true, then the statement under the if expression gets executed, otherwise if the else block is present then the statement of the else expression should be executed. Either the **if** part will get executed or the **else** part will be executed.

Syntax:

```
if (expression)
{
        Statement1;
}

else
{
        Statement2;
}
```

The flowchart for 'if-else' statement is shown below:

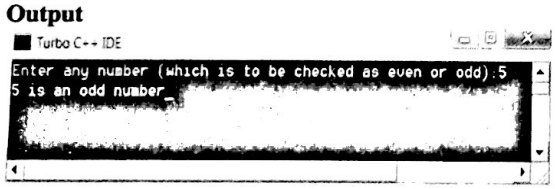


Flow chart of if-else Statement

Program: Illustrating Whether a Number is Even or Odd Using if-else Statement

```
#include<stdio.h>
#include<conio.h>
void main()
{    int n;
     clrscr();
     printf("Enter any number (which is to be checked as even or odd):");
     scanf("%d", &n);
     if (n%2==0)        /* test for even*/
     printf("%d number is even number",n);
     else
     printf("%d is an odd number",n);
     getch();
}
```

Output



Ques 3) Discuss about if-else-if and nested if-else statement with suitable example.

Or

Write the syntax and flowchart for if-else-if and nested if-else statement.

Ans: if-else-if Statement

An **if** statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions using single **if...else if** statement.

Syntax:

```
if(boolean_expression 1)
{
 /* Executes when the boolean expression 1 is true */
}
else if( boolean_expression 2)
{
 /* Executes when the boolean expression 2 is true */
}
else if( boolean_expression 3)
{
 /* Executes when the boolean expression 3 is true */
}
else
{
 /* executes when the none of the above condition is true */
}
```

The flowchart for **if-else-if** statement is shown below:

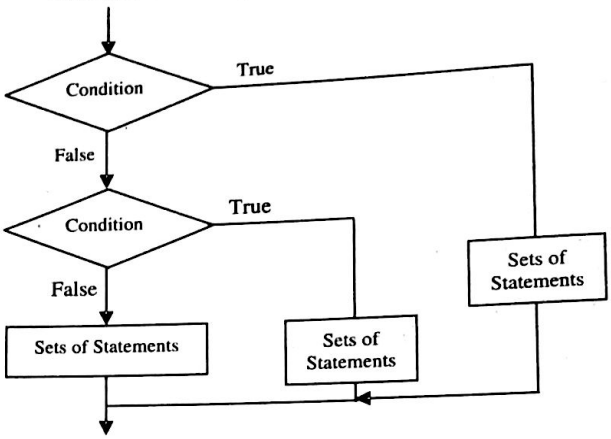
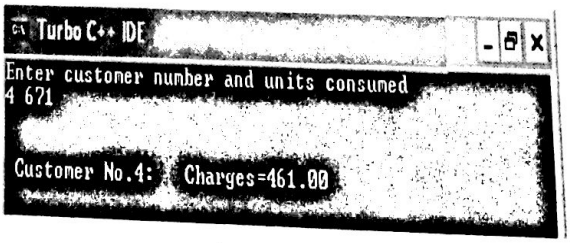


Figure 2.1: Flow Chart Representation of If-Else-If Control Statement

Program: Illustrating if-else-if Statement

```
#include<stdio.h>
#include<conio.h>
void main()
{    int units, custnum;
     float charges;
     clrscr();
     printf("Enter customer number and units consumed\n");
     scanf("%d %d", &custnum, &units);
     if(units<=200)
          charges=0.5*units;
     else if(units<=400)
          charges=100+.65*(units–200);
     else if(units<=600)
          charges=230+0.8*(units–400);
     else
          charges=390+(units–600);
          printf("\n\n Customer No. %d:    Charges=%.2f\n",
          custnum, charges);
     getch();
}
```

Output



Nested if-else Statement

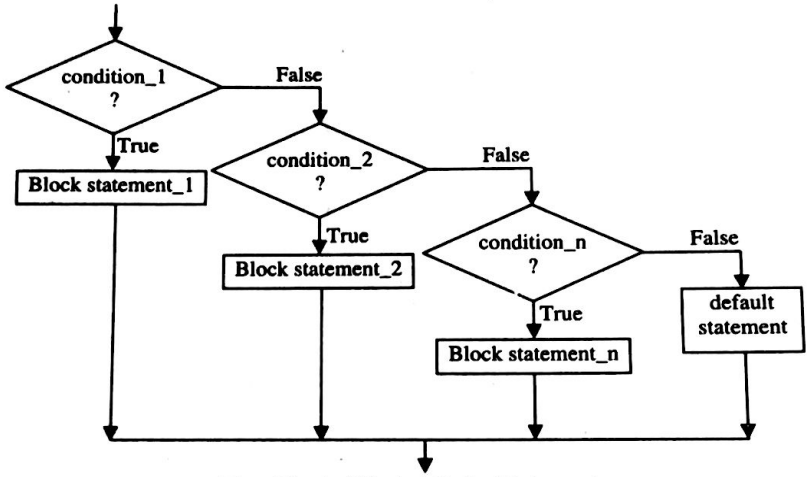
It is a conditional statement which is used when one wants to check more than one condition at a time in a same program. The conditions are executed from top to bottom checking each condition whether it meets the conditional criteria or not.

If it found the condition is true then it executes the block of associated statements of true part else it goes to next condition to execute.

Syntax:

```
if (expression)
{
   if (condition)
   {
      Statements;
   }
   else
   {
      Statements;
   }
}
else
{
   Statement;
}
```

The flowchart for **nested if-else** condition is shown below:

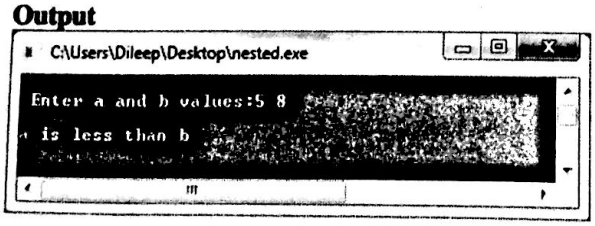


Flow Chart of Nested if-else Statement

Program: Illustrating Nested if-else Statement

```
#include<stdio.h>
#include<conio.h>
int main()
{  int a, b;
   printf("\n Enter a and b values:");
   scanf("%d %d",  &a, &b);
   if(a>b)
   {
     if((a!=0) && (b!=0))
       printf("\n a and b both are +ve and a>b");
     else
       printf("\n a is greater than b only");
   }
   else
     printf("\na is less than b");
   getch();
   return 0;
}
```

Output



Ques 4) Describe about the types of looping statements in C with necessary syntax.

Or

Draw flowchart for do-while loop statement.

Or

Describe iteration and program loops in C language.

Or

Differentiate while and do-while statements.

Ans: Program Loops and Iterations

Iteration is a process wherein a set of instructions or structures are repeated in a sequence a specified number of times or until a condition is met. When the first set of instructions is executed again, it is called an **iteration**. When a sequence of instructions is executed in a repeated manner, it is called a **loop**.

Types of Looping Statements

There are three types of looping statements in C:

1) **for Loop Statement:** Until a particular condition (either pre-determined or open minded) is encountered, a set of instructions are repeated.

   Syntax:

   for(initialization; condition; increment) statement;

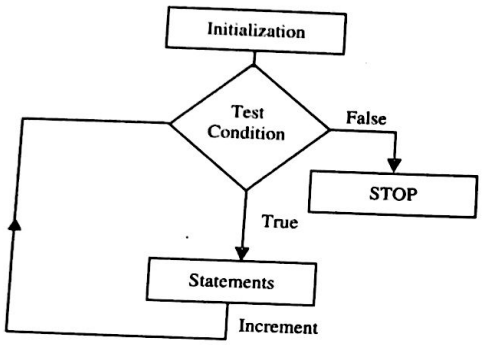
Where,

i) **Initialization:** Before running any loop, the first thing to be done is initializing the variables of the loop.

ii) **Condition:** In next step, the given conditions are tested. If the value of condition is true then the program runs, while the loop exits when the condition is false.

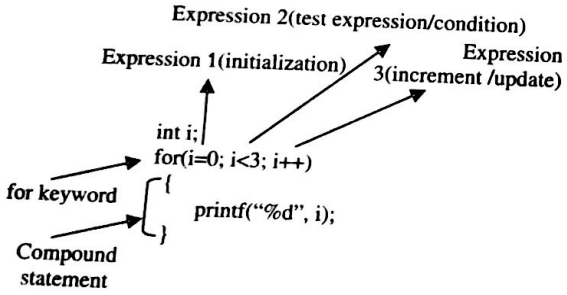
iii) **Increment:** After the completion of every cycle of loop, the increment operation is performed so that next cycle can be run.

The flowchart for the 'for' loop is shown below:



Flow Chart of For Loop

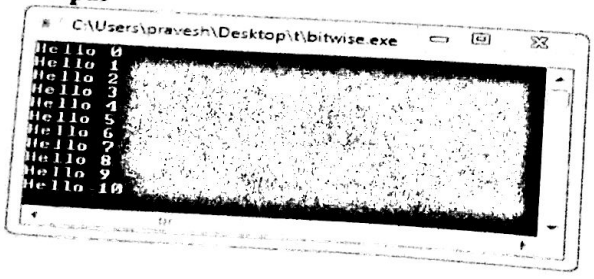
For example, the following code segment illustrates the use of for loop. The 'for' loop is used here to print the numbers whose value is less than 3.



Program : Illustrating 'for' Loop

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

Output

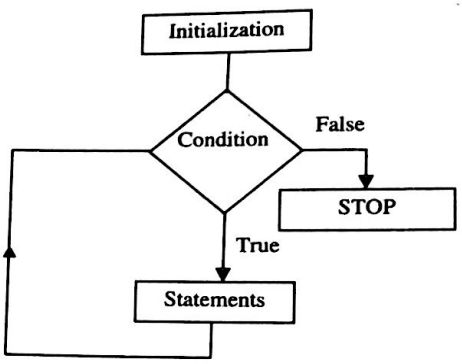


2) **while Loop Statement:** In while loop, the test condition is checked first after that loop body is executed. In case of false condition, the loop will exit. A separate conditional test will not need to be performed before the loop.

Syntax:

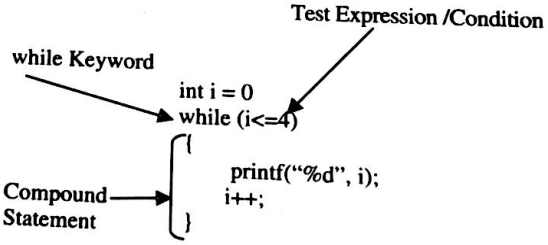
```
while(condition)
{
   statements;
}
```

The flowchart for the 'while' loop is shown below:



Flowchart of While Loop

For example, the following code segment illustrates the use of while loop. The 'while' loop is used here to print a number less than or equal to 4:



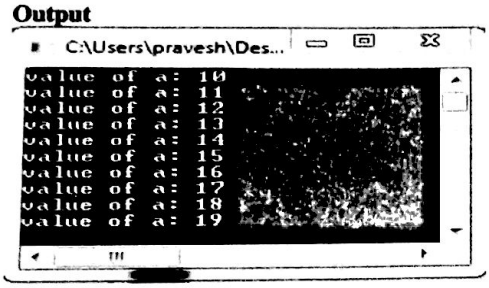
The given statement is either empty statement, single statement, or a block of statements, any expression can be the condition and any non-zero value will be considered true. In case of true value for condition, the loop iterates while for false condition the line of code which is just after the loop gets executed.

For example, let us finding average of a list of numbers. Here, the average of a list of n numbers is calculated by using the while statement. Initially, the partial sum is set to zero and after each new number is being read, the value of the partial sum will be updated. So, for proper execution the problem will have to be solved using the while loop.

Program: Illustrating 'while' Loop

```
#include <stdio.h>
#include <conio.h>
int main ()
{
  /* local variable definition */
  int a = 10;
  /* while loop execution */
  while( a < 20 )
  {
    printf("value of a: %d\n", a);
    a++;
  }
getch(); return 0;
}
```

Output



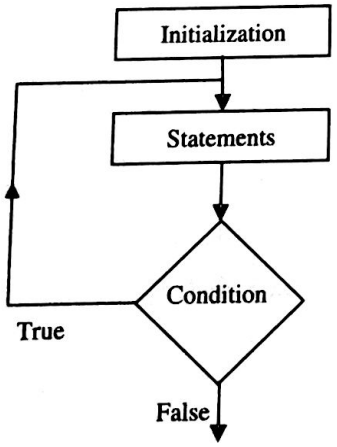
3) **do-while Loop Statement:** In do-while loop the condition is checked at the bottom of the loop. Hence, the execution of the do-while loop will be execute at least once. The syntax of while loop is given below:

Syntax:

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

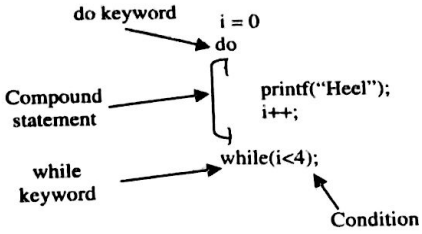
The curly braces are used when more than one statement is present and in case of single statement curly braces are not necessary. But, the curly braces are used so that there is no confusion with the while statement. Till the condition is true the loop will not be terminated.

The flowchart for the 'do-while' loop is shown below:



Flow Chart of do-while Loop

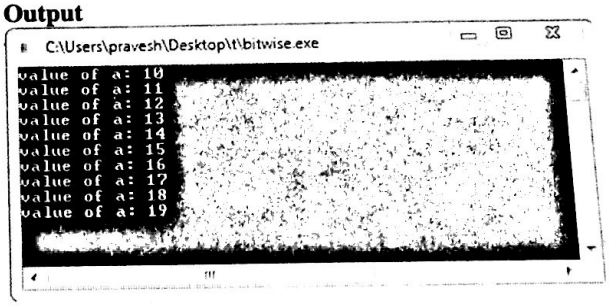
For example, the following code segment illustrates the use of do-while loop. The 'do-while' loop is used for printing Heel thrice:



Program: Illustrating 'do-while' Loop

```
#include <stdio.h>
#include <conio.h>
int main ()
{
  /* local variable definition */
  int a = 10;
  /* do loop execution */
  do
  {
     printf("value of a: %d\n", a);
     a = a + 1;
  }while( a < 20 );
getch(); return 0;
}
```

Output



Difference between while and do-while Statement

While Statement	Do-While Statement
It is a looping construct that will execute only if the test condition is true.	It is a looping construct that will execute at least once even if the test conditions are false.
The general format is: **while(expression)** **statement;** The **expression** is checked at the beginning of the loop. The **statement** will be executed only if the **expression** is satisfied or TRUE.	The general format is: **do** **statement;** **while(expression);** The **expression** is checked at the end of the loop. The **statement** will be executed at least once even if the **expression** is not satisfied or TRUE.
It is an entry-controlled loop.	It is an exit-controlled loop.
It is generally used for implementing common looping situations.	It is typically used for implementing menu-based programs where the menu is required to be printed at least once.

Ques 5) What are the multiple loops variables?

Ans: Multiple Loop Variables

Any field can have more than one expression but each field should be properly separated using commas. **For example,** in the code segment illustrated below:

```
for(i=0, j=0; i<10; ++i)
        ⋮
```

Initial values of i and j are 0. Both expressions i=0 and j=0 are part of init_expression field of loop, but separated by a comma. Another example of for loop is given below:

```
for(i=0, j=100; i<10; ++i, j=j-10)
        ⋮
```

Two index variables i and j are set-up. Before starting the loop value of i is set to 0 and j is set to 100. After completion of every cycle of the loop, i is incremented by 1, i.e., i=i+1 and j is decremented by 10, i.e., j=j–10.

The above illustrated examples include only one variable within the for loop construct. Modification of various variables within the looping process is facilitated by Objective-C. In the given below example, variables i and j of for loop are incremented:

Program: Illustrating Multiple Loop Variables

```
void main()
{
    int i,j;
    for(i=0,j=0;i<5,j<7;i++,j++)
    {
        printf("Value of i=%d\n Value of j=%d",i,j);
    }
}
```

Explanation: Here, i=i+1 and j=j+1, using for loop.

Ques 6) Explain the switch statement with a suitable program. Also write the syntax for switch statement?

Or

Describe the structure of switch-case with neat example.

Or

What is the role of switch statement in C programming language? Explain with example.

Or

Differentiate between the nested.... if and the switch statements in C language with suitable example.

Or

What is the advantage of switch statement over if-else statement?

Ans: Switch Statement

A switch is multiple-branch selection statement which is in-built. The expression's value is cross-checked with an integer list or character constants by the switch. In case of encountering the identical match, the statements linked to constant are executed upto the end of switch or the break

statement. At the occurrence of the break statement, the normal control flow gets interrupted and end of code is encountered, or the normal execution is resumed, hence also known as multi-way decision.

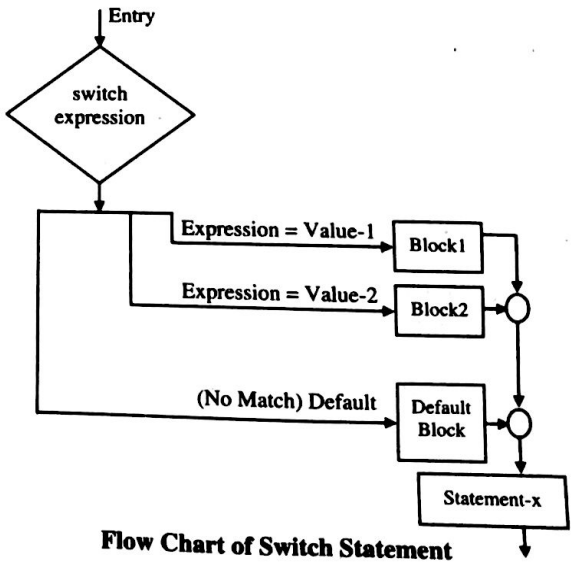
In case of no identical match, the **default** statement gets executed. The default statement may be/may not be present. In absence of default statement and no identical matching, no action takes place.

The switch statement allows the programmer to evaluate an expression and to immediately jump to a specific case lable.

Syntax:

```
switch (expression)
{
case constant1:
    statement sequence
break;
case constant2:
    statement sequence
break;
case constant3:
    statement sequence
break;
.
.
.
default
    statement sequence
}
```

The flowchart for **switch** statement is shown below:



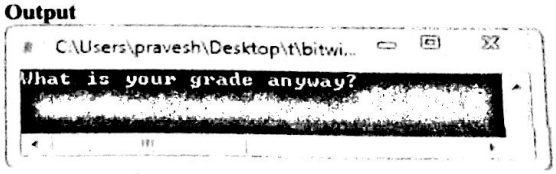
Flow Chart of Switch Statement

Program: Illustrating Switch Statement

```
#include <stdio.h>
#include<conio.h>
main()
{
   int  Grade = 'L';

   switch( Grade )
```

```
{
  case 'A' : printf( "Excellent\n" );
             break;
  case 'B' : printf( "Good\n" );
             break;
  case 'C' : printf( "Fine\n" );
             break;
  case 'D' : printf( "OK\n" );
             break;
  case 'F' : printf( "You must do better than this\n" );
             break;
  default  : printf( "What is your grade anyway?\n" );
             break;
}
getch();
}
```

Output



Difference between Nested if Statement and switch Statement

Nested if/ if else	switch
In nested if..else, the control goes through the every else if statement until it finds true value of the statement or it comes to the end of the else if ladder.	In case of switch case, as per the value of the switch, the control jumps to the corresponding case.
The nested if..else case is less compact than switch. So, nested is considered to be less readable.	The switch case is more compact than lot of nested else if. So, switch is considered to be more readable.
There is no need of use of break in nested if..else statement.	The use of break statement in switch is essential.
Nested if..else is considered to be more flexible.	The switch statement is considered to be less flexible than the else if ladder, because it allows only testing of a single expression against a list of discrete values.
In case of nested if..else, the code needs to be processed in the order determined by the programmer.	Since the compiler is capable of optimizing the switch statement, they are generally considered to be more efficient. Each case in switch statement is independent of the previous one.
Nested if..else works on the basis of true false(zero/non-zero) basis.	Switch case statement work on the basis of equality operator.
Debugging is difficult.	It is easy in debugging.
Maintenance is not easy.	Its maintainability is easy.

Ques 7) Explain the various unconditional type of control statement.

Or

Write a short note on GOTO statement. Also discuss the use of label in GOTO statement.

Or

What is the difference between break and continue in C?

Ans: Unconditional Type of Control Statement

There are three basic types of unconditional control statements:

1) **break Statement:** break statement is a simple statement. Break statement is useful only when present in the switch, do-while or for statement. The flow control swaps to the statement located just after the body of the statement containing the break, as the break statement gets executed.

The break statements plays vital role in switch statements as it provides the flexibility to shift the control as per requirement. The validity of break statement usage within loop is uncertain, especially when the circumstances are not normal and the loop becomes uncontrollable.

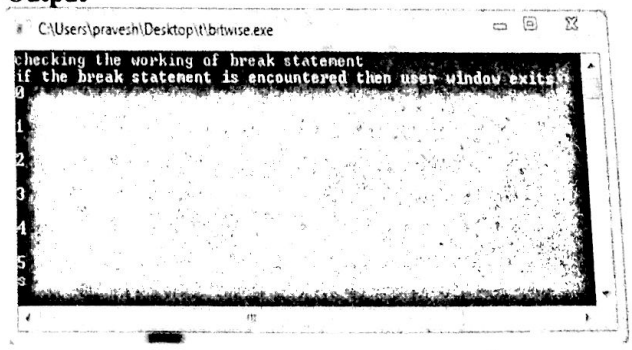
Syntax:

break;

Program: Illustrating Break Statement

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{   int i;
    printf("checking the working of break statement\n");
    printf("if the break statement is encountered then
    user window exits");
    for(i = 0; i<100; i++)
    {   if(getchar() == 's')
            break;
            printf("%d\n", i);}
    exit(0);
    getch();}
```

Output



Explanation: The inputs of the program are arranged sequentially, and before giving the next input for printing, each input character is read. The loop will be exited if 's' is typed.

2) **continue Statement:** The usage of continue statement is not as vital as others. It is not used with the switch statements and rest usage rules are the same as that of break.

Syntax:

```
continue;
```

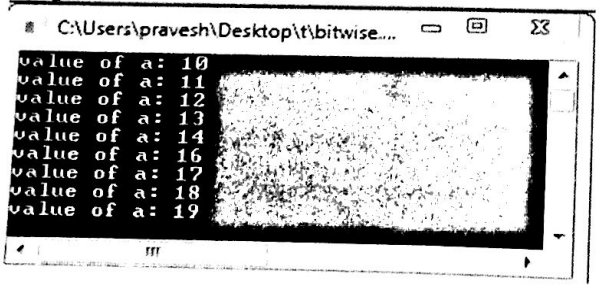
As soon as the continue statement gets executed, the next available (smallest) loop (i.e., do, while or for statement) is initiated.

The continue statement is present at the bottom of the loop so that rest body of the loop is executed atleast once. The continue statement induces one more indentation level to enhance the readability in case of handling the large for loops.

Program: Illustrating continue Statement

```
#include<stdio.h>
#include<conio.h>
int main ()
{
 /* local variable definition */
 int a = 10;
 /* do loop execution */
 do
 {
  if( a == 15)
  {
   /* skip the iteration */
   a = a + 1;
   continue;
  }
  printf("value of a: %d\n", a);
  a++;
  }while( a < 20 );

getch(); return 0;
}
```

Output



Difference between break and continue Statement

Table: Break Statement *versus* Continue Statement

	break Statement	continue Statement
1)	Switch statement uses break statement.	Continue statements are not for switch statements.
2)	Can exit the loop without even being completed.	Statements can be skipped present in the same loop.
3)	After completion of the loop, the control is transferred to the statements which are present after the loop.	After loop termination, the control does not goes out instead is transferred back to the loop.
4)	It is not necessary that the loop will complete whole execution (intended number of iterations).	The intended numbers of iterations of the loop are completed.

3) **goto Statements:** goto statement transfers the control to some other part of the program, hence, used for altering the program execution sequence.

Syntax:

```
goto label;
...
...
label: statement;
```

Here **label** can be any plain text except C keyword and it can be set anywhere in the C program above or below to **goto** statement. The **label** is an identifier. When goto statement is encountered, control of the program jumps to label: and starts executing the code.

Figure 2.2 shows the jumping of goto statement:

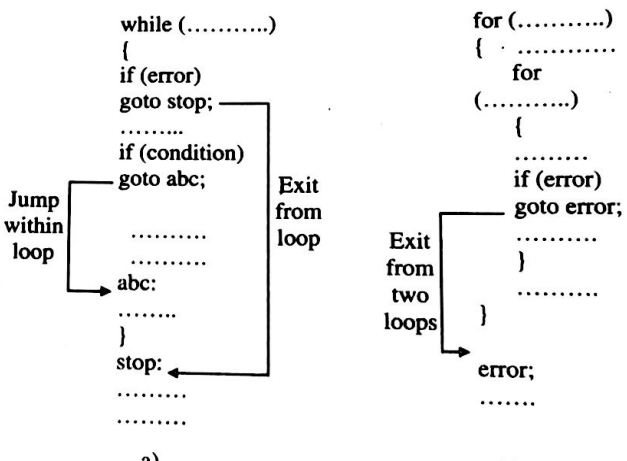


Figure 2.2: Jumping with goto Statement

An identifier ending with a colon is a label. Each label is assigned a unique name, hence, the ambiguity of the names in variables or functions are reduced.

The name space of a label is for that particular function in which it is present, hence, for other functions the same label name can be used. If a label is mentioned in a goto statement then it can be used even before being declared.

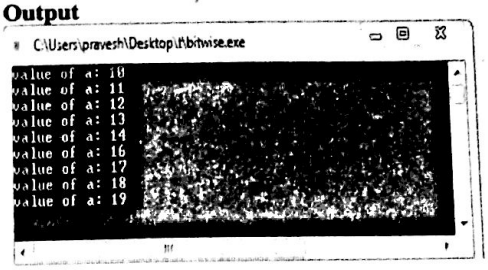
Though labels are empty, then also it should be part of a full statement, the care should be taken if the label is to be placed at a compound statement's end.

If one wants to move from one function to another function then it is not possible as the label name can be seen within own function not the other.

Program: /*Illustrating goto Statement*/

```
#include <stdio.h>
#include<conio.h>
int main ()
{
/* local variable definition */
  int a = 10;
  /* do loop execution */
  LOOP:do
  {
    if( a == 15)
    {
      /* skip the iteration */
      a = a + 1;
      goto LOOP;
    }
    printf("value of a: %d\n", a);
    a++;

  }while( a < 20 );
   getch();
}
```

Output



PROGRAMMING EXAMPLES

Ques 8) Give the loop statement to print the following sequence of integer.
–6 –4 –2 0 2 4 6;

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a = –6;
do
{
printf("%d\t", a);
   a=a+2;
 }
  while( a < 8 );
getch();
}
```

Explanation: In the above program, variable 'a' is initialised with value '–6' and a 'do while' loop is applied in which the equation a= a+2 gives the result '–6 –4 –2 0 2 4 6' because 'a' gets incremented with +2 value. The loop is executed till a<6.

Ques 9) Write a program in C to print the following pattern:

```
A B C D E F G F E D C B A
A B C D E F   F E D C B A
A B C D E       E D C B A
A B C D           D C B A
A B C               C B A
A B                   B A
A                       A
```

Ans:

```
#include<stdio.h>
 int main()
{
  int i,j,k,l,m;
 for(i=0;i<=6;i++)
 {
   for(k=65;k<=71-i;k++)
      printf("%c",k);

   for(j=1;j<=i*2–1;j++)
      printf(" ");

   for(l=71–i;l>=65;l – –)
      if(l!=71)
         printf("%c",l);

   printf("\n");
 }
  return 0;
}
```

Ques 10) Write a program in C language to find the sum of series.

Sum = $x/1! - x^3/3! + x^5/5! - x^7/7! + x^9/9!$

Ans: C Program to Find the Sum of Series

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
int main()
{
    int i,n,x,a,count,j;

    float sum,b;
    printf("Enter the limit of the series: ");
    scanf("%d",&n);

    printf("Enter the value of x: ");
    scanf("%d",&x);
    printf("series is:\t");
    count=0;
    sum=0;
    for(i=1;i<=2*n;i=i+2)
    {
            count++;
            a=pow(x,i);
```

```
        b=1;
        for(j=1;j<=i;j++)
            b=b*j;
        printf("%d/%d!", a,i);
        if(i<2*n-1)
            {
                if(count%2==0)
                    printf(" + ");
                else
                    printf(" - ");
            }
        if(i<2*n)
            {
                if(count%2==0)
                    sum=sum-(a/b);
                else
                    sum=sum+(a/b);}}
    printf("\nSum of series is: %f", sum);
    getch();
    return(0); }
```

Ques 11) Write a program in C to generate a following numbers structure:

```
        1
      1 2
    1 2 3
  1 2 3 4
1 2 3 4 5
```

Ans: Program in C to Generate Numbers Structure

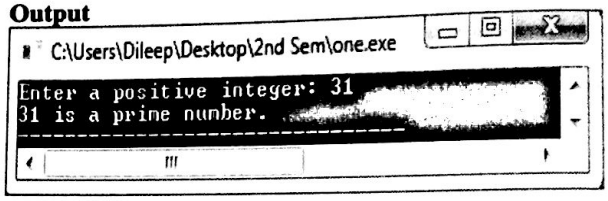
```
#include<stdio.h>
#include<conio.h>
int main()
{
 int num,r,c,sp;
 printf("Enter loop repeat number(rows): ");
 scanf("%d",&num);
 for(r=1; num>=r; r++)
 {
 for(sp=num-r; sp>=1; sp--)
   printf(" ");
 for(c=1; c<=r; c++)
   printf("%d",c);
 printf("\n");
 }
 getch();
 return 0;
}
```

Ques 12) Write a program in C to check whether a given integer is prime or not?

Ans: C Program to Check Whether a Given Integer is Prime or Not

```
#include <stdio.h>
int main()
{
  int n, i, flag = 0;
  printf("Enter a positive integer: ");
  scanf("%d",&n);
  for(i=2; i<=n/2; ++i)
  { // condition for nonprime number
    if(n%i==0)
    {
      flag=1;
      break;
    } }

  if (flag==0)
    printf("%d is a prime number.",n);
  else
    printf("%d is not a prime number.",n);

  return 0;}
```

Output



Explanation: If the **for** loop terminates when the test expression of loop **i <= n/2** is false, the entered number is a prime number. The value of **flag** is equal to 0 in this case. If the loop terminates because of **break** statement inside the **if** statement, the entered number is a nonprime number. The value of flag is 1 in this case.

Ques 13) Write a program in C to display all the prime number between 100 and 1000.

Ans: C Program to Display all the Prime Number between 100 and 1000

```
#include <stdio.h>
int main()
{
  int n1, n2, i, flag;
  printf("Enter two numbers(intevals): ");
  scanf("%d %d", &n1, &n2);
  printf("Prime numbers between %d and %d are: ", n1, n2);
  while (n1 < n2)
  {
    flag=0;
    for(i=2; i<=n1/2; ++i)
    {
      if(n1%i == 0)
      {
        flag=1;
        break;
      }}
    if (flag == 0)
      printf("%d ",n1);
    ++n1; }
  return 0;}
```

Ques 14) Write a program in C to read a five digit number if it is even then add up the digits otherwise multiply them and print the result.

Ans:

```
#include <stdio.h>
int main()
{
   int number,remainder, result = 0, mul = 1;

   printf("Enter an number: ");
   scanf("%d",&number);

   // True if the number is perfectly divisible by 2
   if(number%2 == 0)
     {printf("%d is even.", number);
     while (number != 0)
          {
     remainder = number%10;
     result += remainder;
     number /= 10;
          }
          printf("\nSum of the number is: %d\n", result);
    }
   else
     {printf("%d is odd.", number);
     while (number != 0)
          {
     remainder = number%10;
     mul = remainder*mul;
     number /= 10;
          }
         printf("\n Multiply of the number is: %d\n", mul);
    }
  return 0;
}
```

Ques 15) Write a program in C using switch statement to find the value of Y for a given value of N between 1 to 4 :

If N = 1	$Y = (ax + b)^2$
If N = 2	$Y = ax^2 + b^3$
If N = 3	$Y = -ax + b$
If N = 4	$Y = a^2 + x$

Ans:

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

  int N;
  printf("Enter the value of N between 1 to 4:");
  scanf("%d", &N);

 switch(N)
 {
   case 1:
     printf("Y=(ax+b)^2");
     break;
   case 2:
     printf("Y= ax^2 + b^3");
     break;
   case 3:
     printf("Y= - ax + b");
     break;
   case 4:
     printf("Y= a^2 + x");
     break;
  }
 getch();
  return 0;
}
```

Ques 16) Write a program in C language to generate the given series upto term less than 200.

1 – 4 + 9 – 16 + 25 – .....

Ans:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int i;

        for(i=1;i<=200;i++)
        {

          if(i%2 == 0)
              printf("\t-%d",i*i);
           else
                printf("\t%d",i*i);
        }
        getch();
        return 0;
}
```

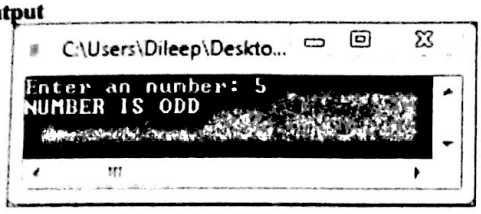
Ques 17) Write a program in C to determine whether a given number is 'odd' or 'even' and print the message:
NUMBER IS EVEN
or
NUMBER IS ODD
without using else option.

Ans: C Program to Determine Whether a Given Number is 'Odd' or 'Even' and Print the Message Without Using else Option

```
#include <stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
   int number,remainder, result = 0, mul = 1;

   printf("Enter an number: ");
   scanf("%d",&number);

   // True if the number is perfectly divisible by 2
   if(number%2 == 0)
     {printf("NUMBER IS EVEN");
```

```
    getch();
        return 0;
  exit(0);
    }
  printf("NUMBER IS ODD");
   getch();
  return 0;}
```

Output



Ques 18) Write a program in C to read the age of 100 persons and count the number of persons in the age group 50 to 60. Use for and continue statements.

Ans: C Program to Read the Age of 100 Persons and Count the Number of Persons in the Age Group 50 to 60

```
#include<stdio.h>
#include<conio.h>
int main()
{  int count, i, age;
  count=0;printf("Input age of 100 persons:");
  for(i=1;i<=100;i++)
  {
     scanf("%d", &age);
     if(age>=50&&age<=60)
       count+=1;
       continue;
  }
  printf("the countable number is:%d", count);
  getch();
  return 0;
}
```

Ques 19) Write a program to reverse a given number.

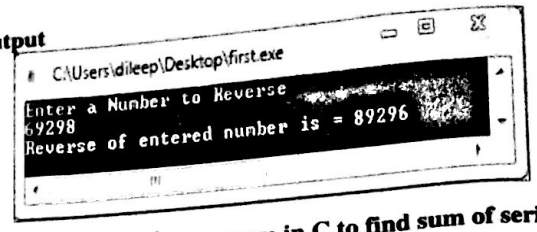
Or

Make a program in C to reverse digits of a number.

Ans: Program to Reverse Given Number

```
#include<stdio.h>
#include<conio.h>
int main()
{  int n, reverse = 0;
  printf("Enter a Number to Reverse\n");
  scanf("%d", &n);
  while (n != 0)
  {
    reverse = reverse * 10;
    reverse = reverse + n%10;
    n = n/10;
  }
  printf("Reverse of entered number is = %d\n", reverse);
  return 0;
  getch();
}
```

Output

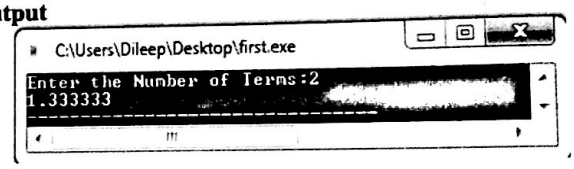


Ques 20) Write a program in C to find sum of series $S = 2 - \frac{2}{3} + \frac{2}{5} - \frac{2}{7} + ... + \frac{2}{n}$, where n is always odd.

Ans: Program for Sum of Series

```
#include <stdio.h>

int main() {
    int i,n;
    double p = 0;
printf("Enter the Number of Terms:");
scanf("%d", &n);
for ( i = 1; i <= n; i++) {
    if ( i % 2 == 0) {
        p = p - (2.0 / (2 * i - 1)); }
else {
    p = p + (2.0 / (2 * i - 1)); }}
    printf( "%f\t", p );
    return 0; }
```

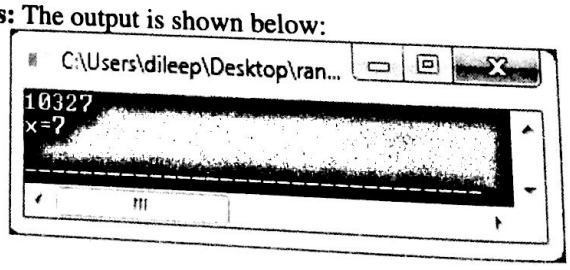
Output



Ques 21) Find the output of following syntactically correct code and explain how you obtained it:

```
#include <stdio.h>
main()
{
    int i=0, x=0;
    for (i=1; i<10; ++i)
    {   if (i%2==1)
            x+=i;
        else
            x--;
        printf ("%d",x);
        if ((i+x)%4==0)
            break;
    }
    printf("\nx=%d\n",x);
}
```

Ans: The output is shown below:



Explanation: In this program, the initial values of x and i are zeros. In the for loop the value of i is assigned 1, which is less than 10 and checked by if statement. Since i%2 gives 1 which satisfies the condition, hence value of x is increased by zero to one by statement x =x + i (= 0+1). This value is printed on screen. Value of i is increased to 2. Now if condition fails and else condition changes the value from 1 to zero by x – – condition. This value is printed on screen as zero. Now again other if condition is checked which is failed, then break condition does not executes.

Now values of i is 3 and x is 0. First if condition satisfied as 3%2 =1.Hence the x= 0 +3 =3. This value is printed on screen. Here the other if condition fails because condition ((6%4≠0)) does not satisfied.

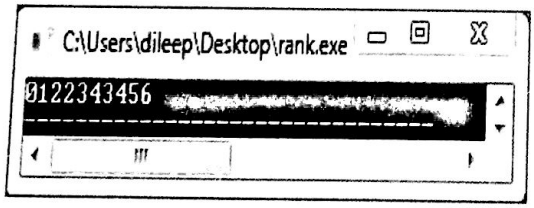
Now values of i is 4 and x is 3. Then if condition (4%2 =0) now fails and else condition is executed. The value of x is now 7. This value is printed on the screen. Now the second if condition also does not satisfies. Now the second if condition ((7+5)%4 = = 0) is satisfied. Now the break statement terminates the loop. And printf() function outside the loop prints the value of x =7 on the screen.

The control again goes in for loop with value i = 5 and x =2. If condition (5%2 ==1) now again satisfied. Hence value of x = 2 + 5 = 7. This value is again printed on screen.

Ques 22) Find the output of following syntactically correct code:

```
#include <stdio.h>
main()
{
    int i,j,k,x=0;
    for (i=0; i<5; ++i)
        for (j=0; j<i; j++)
        {
            k=(i+j–1);
            printf("%d", k);
        }
}
```

Ans: The output is shown below:



Ques 23) Write a C program which will find sum of following series using loops:

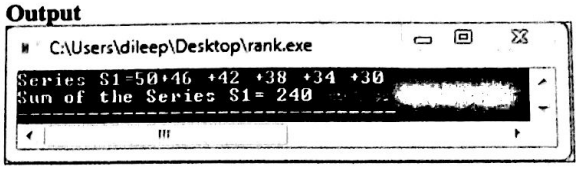
S1=50+46+42+...upto m1 (e.g., upto 30)

S2=47+45+43+...m2 values (e.g., 4 values, i.e., 47+45+43+41)

Ans: C Program for Series S1

```
#include<stdio.h>
void main()
{
  int val=50,i;
  int sum=0;
  printf("Series S1=");
  if(val==50) {
    printf("%d",val);
    sum =sum + val;
    val =val-4; }
  while (val>=30) {
    printf("+%d ",val);
    sum=sum + val;
    val= val-4;
  }
 printf("\nSum of the Series S1= %d",sum);
}
```

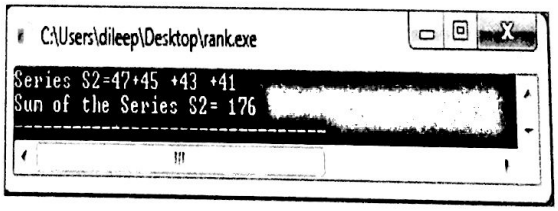
Output



C Program for Series S2

```
#include<stdio.h>
void main()
{  int val=47, i;
   int sum=0;
     printf("Series S2=");
     if(val==47) {
          printf("%d",val);
          sum =sum + val;
          val =val-2; }
   while (val>=41)
   {  printf("+%d ",val);
     sum=sum + val;
     val= val-2;
   }
 printf("\nSum of the Series S2= %d",sum);
}
```

Output



Ques 24) How many times while loop executes? Justify. Also specify the value of a after every iteration of the loop:

```
int x=65, y=45, a=1;
while (a<9)
{
    if(x>y)
    { x=x–5; a=a –2;}
    a=a+1;
}
```

Ans: The loop will execute 16 times until the value of a will be 9. The if condition will execute only 3 times as x will be equal to y after 4th iteration.

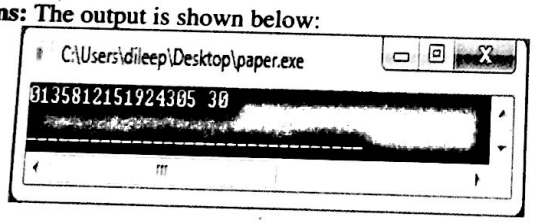
The value of x, y and a will be as follows.
x = 60, y = 45, a = 0
x = 55, y = 45, a = −1
x = 50, y = 45, a = −2
x = 45, y = 45, a = −3
x = 45, y = 45, a = −2
x = 45, y = 45, a = −1
x = 45, y = 45, a = 0
x = 45, y = 45, a = 1
x = 45, y = 45, a = 2
x = 45, y = 45, a = 3
x = 45, y = 45, a = 4
x = 45, y = 45, a = 5
x = 45, y = 45, a = 6
x = 45, y = 45, a = 7
x = 45, y = 45, a = 8
x = 45, y = 45, a = 9

Ques 25) Find the output of following syntactically correct code and justify:

```
int a, b, x=0;
for (a=0; a<5; a=a+1)
for(b=0; b<a; b=b+1)
{   x=x + (a + b–1);
    printf ("%d", x);
}
printf("%d %d\n", a, x);
```

If { } are removed from the above code, what will be the output? Justify.

Ans: The output is shown below:



Explanation:
In 1st Pass x=0
In 2nd Pass x=1
In 3rd Pass x=3
In 4th Pass x=5
In 5th Pass x=8
In 6th Pass x=12
In 7th Pass x=15
In 8th Pass x=19
In 9th Pass x=24
In 10th Pass x=30

After 10th pass, iteration loop will not execute as value of a will be 5 and condition of outer loop will be false. The outer printf function will print a =5 and x=30.

Removing curly braces will affect the iteration and print the value of first x = 30 and then by second printf as a =5 and x =30.

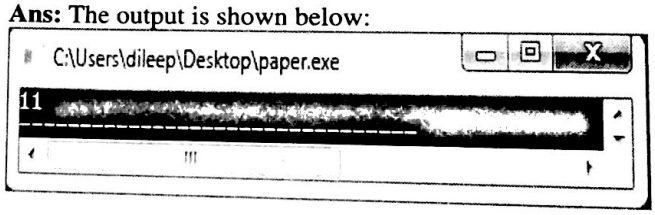
Ques 26) Write the output of the following code:

i)
```
int i;
for(i=0, i≤10, i– –)
    printf("%d", i);
```

Ans: Error message will occur as ';' is required in for loop in place of ')'.

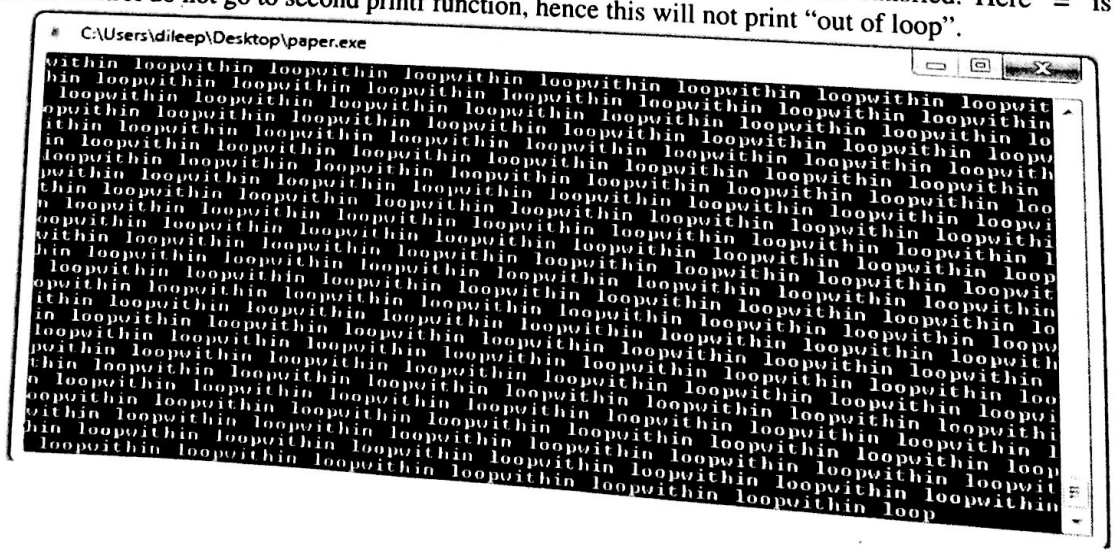
ii)
```
int i;
for(i=0; i≤10; i++);
    printf ("%d", i);
```

Ans: The output is shown below:



iii)
```
int x=10;
while (x=5)
printf ("within loop");
printf ("out of loop");
```

Ans: This program prints the "within loop" infinite times as the x=5 conditions are satisfied. Here '=' is assignment operator. As the control do not go to second printf function, hence this will not print "out of loop".

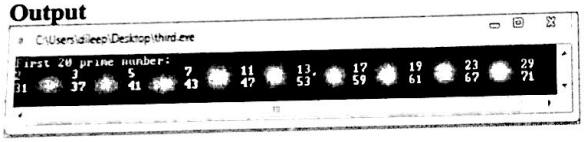


Ques 27) Write a program to display first 20 prime numbers.

Ans: Program to Display First 20 Prime Numbers

```
#include<stdio.h>
#include<conio.h>
int main(){
    int n=20,num,t,div,count;
    printf("First 20 prime number: ");
    printf("\n%d\t",2);
    count=1;
    num=3;
    while(count<n)
    {   t=sqrt(num);
        div=2;
        while(div<=t)
        {   if(num%div==0)
            break;
            div++;}
        if(div>t)
        {   printf("%d\t",num);
            count++; }
        num=num+2; }
    getch();
    return 0; }
```

Output



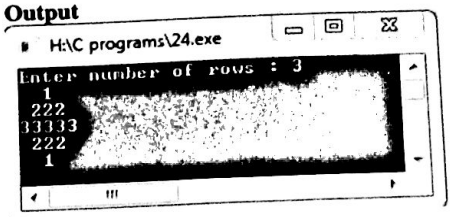
Ques 28) Write a program in C to display the following pattern in C language:

```
        1
    2   2   2
3   3   3   3   3
    2   2   2
        1
```

Ans: C Program to Display Pattern

```
#include<stdio.h>
#include<conio.h>
int main()
{   int num,r,c,sp;
    printf("Enter number of rows : ");
    scanf("%d",&num);
    for(r=1; r<=num; r++)
    {   for(sp=num-r; sp>=1; sp--)
            printf(" ");
        for(c=1; c<=r; c++)
            printf("%d",r);
        for(c=r-1; c>=1; c--)
            printf("%d",r);
        printf("\n"); }
    for(r=1; r<=num; r++)
    {   for(sp=r; sp>=1; sp--)
            printf(" ");
        for(c=1; c<=(num-r); c++)
            printf("%d",num-r);
        for(c=num-r-1; c>=1; c--)
            printf("%d",num-r);
        printf("\n"); }
    getch();
    return 0;}
```

Output

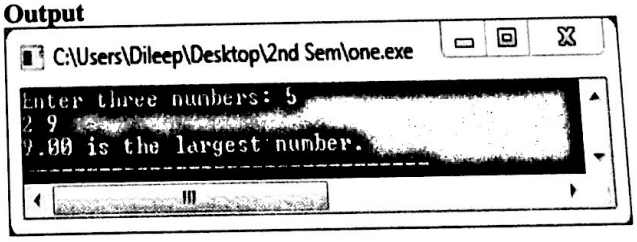


Ques 29) Write a program to find the largest number among three numbers using nested if-else statement.

Ans: C Program to Find the Largest Number Among Three Numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{
  double n1, n2, n3;

  printf("Enter three numbers: ");
  scanf("%lf %lf %lf", &n1, &n2, &n3);

  if( n1>=n2 && n1>=n3)
    printf("%.2lf is the largest number.", n1);

  else if (n2>=n1 && n2>=n3)
    printf("%.2lf is the largest number.", n2);

  else
    printf("%.2lf is the largest number.", n3);

  return 0;
}
```

Output



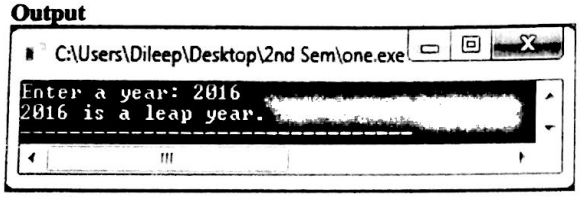
Ques 30) Write a program to check leap year.

Ans: Program to Checking Whether Year is Leap or Not

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

```
int year;
   printf("Enter a year: ");
   scanf("%d",&year);
   if(year%4 == 0)
   {
      if( year%100 == 0)
      {
        // year is divisible by 400, hence the year is a leap year
         if ( year%400 == 0)
            printf("%d is a leap year.", year);
         else
            printf("%d is not a leap year.", year);
      }
      else
         printf("%d is a leap year.", year );
   }
   else
      printf("%d is not a leap year.", year);

   return 0;
}
```

Output

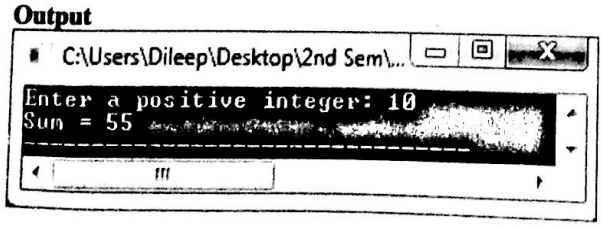


Ques 31) Write a program to calculate the sum of natural numbers using do-while loop.

Ans: Program for Calcuating the Sum of Natural Numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{   int n, i, sum = 0;

   do {
      printf("Enter a positive integer: ");
      scanf("%d",&n);
   }
   while (n <= 0);

   for(i=1; i <= n; ++i)
   {
     sum += i;   // sum = sum+i;
   }
   printf("Sum = %d",sum);
   return 0;
}
```

Output

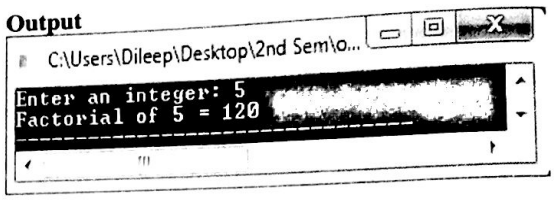


Ques 32) Write a program to find the factorial of a number.

Ans: Program to Find the Factorial of a Number

```
#include<stdio.h>
#include<conio.h>
int main()
{   int n, i;
   unsigned long long factorial = 1;
   printf("Enter an integer: ");
   scanf("%d",&n);
   // show error if the user enters a negative integer
   if (n < 0)
printf("Error! Factorial of a negative number doesn't exist.");
   else
   {for(i=1; i<=n; ++i)
      {  factorial *= i;              // factorial = factorial*i;
      } printf("Factorial of %d = %llu", n, factorial);
   }   return 0;}
```

Output

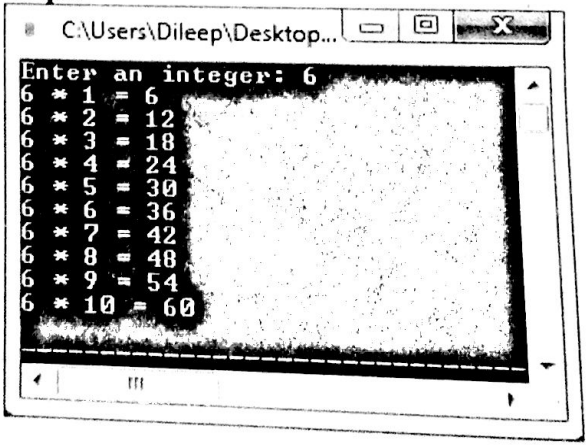


Ques 33) Write a program to find the multiplication table of a given number.

Ans: Program to Display Multiplication Table

```
#include<stdio.h>
#include<conio.h>
int main()
{   int n, i;
   printf("Enter an integer: ");
   scanf("%d",&n);
   for(i=1; i<=10; ++i)
   {  printf("%d * %d = %d \n", n, i, n*i);
   }       return 0;

}
```

Output

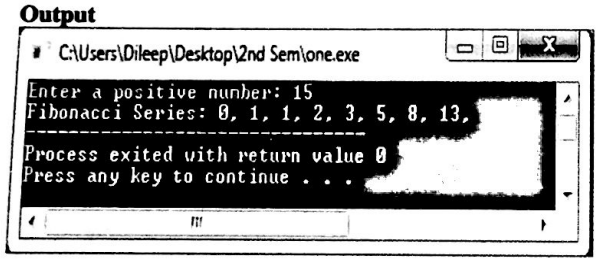


Ques 34) Write a program to generate fibonacci sequence up to a certain number using while loop.

Ans: Program Generating Fibonacci Sequence upto a Certain Number

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int t1 = 0, t2 = 1, nextTerm = 0, n;
   printf("Enter a positive number: ");
   scanf("%d", &n);
   // displays the first two terms which is always 0 and 1
   printf("Fibonacci Series: %d, %d, ", t1, t2);
   nextTerm = t1 + t2;
   while(nextTerm <= n)
   {
      printf("%d, ",nextTerm);
      t1 = t2;
      t2 = nextTerm;
      nextTerm = t1 + t2;
   }

   return 0;
}
```

Output

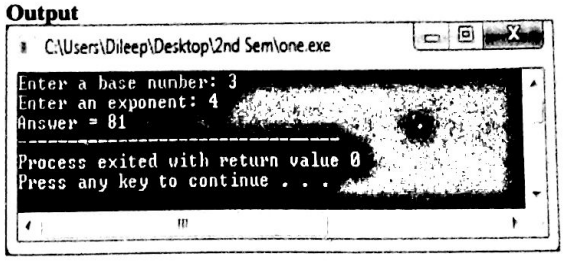


Ques 35) Write a C program to calculate the power of a number.

Ans: C Program to Calculate the Power of a Number

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int base, exponent;
   long long result = 1;

   printf("Enter a base number: ");
   scanf("%d", &base);

   printf("Enter an exponent: ");
   scanf("%d", &exponent);

   while (exponent != 0)
   {
      result *= base;
      --exponent;
   }

   printf("Answer = %lld", result);
   return 0;
}
```

Output

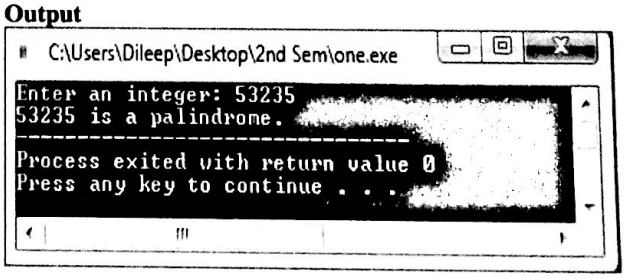


Ques 36) Write a program to check whether a number is palindrome or not.

Ans: Program for Checking Palindrome of a Number

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int n, reversedInteger = 0, remainder, originalInteger;
   printf("Enter an integer: ");
   scanf("%d", &n);
   originalInteger = n;
   // reversed integer is stored in variable
   while( n!=0 )
   {
      remainder = n%10;
      reversedInteger = reversedInteger*10 + remainder;
      n /= 10;
   }
 // palindrome if orignalInteger and reversedInteger are equal
   if (originalInteger == reversedInteger)
      printf("%d is a palindrome.", originalInteger);
   else
      printf("%d is not a palindrome.", originalInteger);

   return 0;
}
```

Output



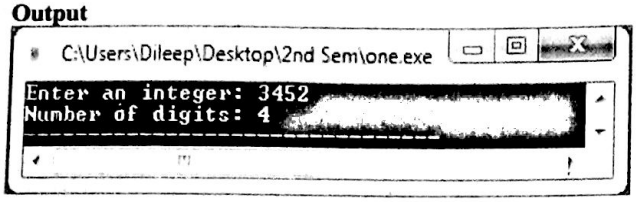
Ques 37) Write a program to count number of digits in an integer.

Ans: Program for Counting Number of Digits in an Integer

```
#include<stdio.h>
#include<conio.h>
int main()
{
   long long n;
   int count = 0;
```

```
    printf("Enter an integer: ");
    scanf("%lld", &n);

    while(n != 0)
    {
      // n = n/10
      n /= 10;
      ++count;
    }

    printf("Number of digits: %d", count);
}
```

Output



Explanation: The integer entered by the user is stored in variable n. Then the while loop is iterated until the test expression n! = 0 is evaluated to 0 (false).

1) After first iteration, the value of n will be 345 and the count is incremented to 1.
2) After second iteration, the value of n will be 34 and the count is incremented to 2.
3) After third iteration, the value of n will be 3 and the count is incremented to 3.
4) After fourth iteration, the value of n will be 0 and the count is incremented to 4.
5) Then the test expression is evaluated to false and the loop terminates.

Ques 38) Write a program to check Armstrong number.

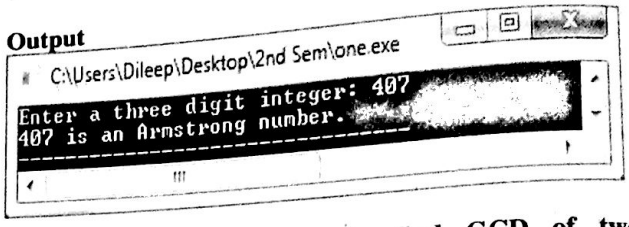
Ans: Program to Checking Whether Number is Armstrong or Not

Armstrong number is a number which is equal to sum of digits raise to the power total number of digits in the number. Some Armstrong numbers are: 0, 1, 2, 3, 153, 370, 407, 1634, 8208 etc.

The program for checking Armstrong number is shown below:

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int number, originalNumber, remainder, result = 0;

  printf("Enter a three digit integer: ");
  scanf("%d", &number);

  originalNumber = number;

  while (originalNumber != 0)
  {
    remainder = originalNumber%10;
    result += remainder*remainder*remainder;
    originalNumber /= 10;
  }
  if(result == number)
    printf("%d is an Armstrong number.",number);
  else
    printf("%d is not an Armstrong number.",number);
  return 0;
}
```

Output



Ques 39) Write a program to find GCD of two numbers using while loop and if...else statement.

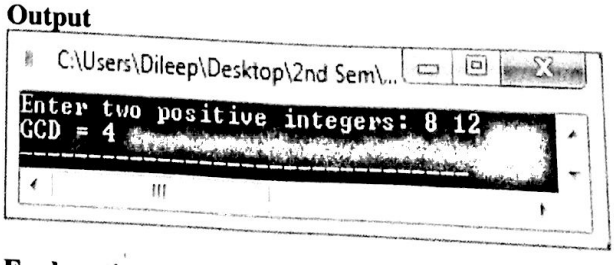
Ans: C Program to Find GCD(Greatest Common Divisor)

The Greatest Common Divisor (GCD) of two or more integers, when at least one of them is not zero, is the largest positive integer that divides the numbers without a remainder. **For example,** the GCD of 8 and 12 is 4.

The **progarm below** illustrates the GCD of two numbers:

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int n1, n2;

  printf("Enter two positive integers: ");
  scanf("%d %d",&n1,&n2);

  while(n1!=n2)
  {
    if(n1 > n2)
      n1 -= n2;
    else
      n2 -= n1;
  }
  printf("GCD = %d",n1);

  return 0;
}
```

Output



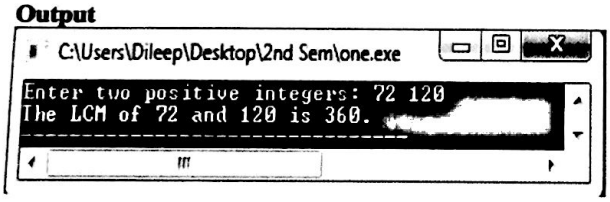
Explanation: In this method, smaller integer is subtracted from the larger integer, and the result is assigned to the variable holding larger integer. This process is continued until n1 and n2 are equal.

Ques 40) Write a program to find the LCM of two numbers.

Ans: C Program to Find the LCM of Two Numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int n1, n2, minMultiple;
   printf("Enter two positive integers: ");
   scanf("%d %d", &n1, &n2);

   // maximum number between n1 and n2 is stored in
   minMultiple
   minMultiple = (n1>n2) ? n1 : n2;

   // Always true
   while(1)
   {
      if( minMultiple%n1==0 && minMultiple%n2==0 )
      {
      printf("The LCM of %d and %d is %d.", n1,
      n2,minMultiple);
         break;
      }
      ++minMultiple;
   }
   return 0;
}
```

Output



Explanation: In this program, the integers entered by the user are stored in variable **n1** and **n2** respectively. The largest number among **n1** and **n2** is stored in **minMultiple**. The LCM of two numbers cannot be less than **minMultiple**.

The test expression of while loop is always true (1). In each iteration, whether **minMultiple** is perfectly divisible by **n1** and **n2** is checked. If this test condition is not true, minMultiple is incremented by 1 and the iteration continues until the test expression of if statement is true.

The LCM of two numbers can also be found using the formula:
LCM = (num1*num2)/GCD

Ques 41) Write a program to print Floyd's triangle.

```
1
2   3
4   5   6
7   8   9   10
11  12  13  14  15
```

Ans: Program Printing Floyd's Triangle

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int rows, i, j, number= 1;

   printf("Enter number of rows: ");
   scanf("%d",&rows);

   for(i=1; i <= rows; i++)
   {
      for(j=1; j <= i; ++j)
      {
         printf("%d ", number);
         ++number;
      }

      printf("\n");
   }

   return 0;
}
```

Output