PROBLEM SOLVING AND PROGRAMMING LAB MANUAL

B. Tech., CSE – I Year I Sem

ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES

(Affiliated to JNTU Anantapur and Approved by AICTE)

Itikalapalli, Ananthapuramu.

Index

Preface

C is a general-purpose high level language that was originally developed by Dennis Ritchie for the UNIX operating system. It was first implemented on the Digital Equipment Corporation PDP-11 computer in 1972. The UNIX operating system and virtually all UNIX applications are written in the C language. C has now become a widely used professional language for various reasons. Easy to learn, structured language, it produces efficient Programs, it can handle low-level activities. It can be compiled on a variety of computers.

Facts about C

C was invented to write an operating system called UNIX. C is a successor of B language which was introduced around 1970. The language was formalized in 1988 by the American National Standard Institute (ANSI). By 1973 UNIX OS almost totally written in C. Today C is the most widely used System Programming Language. Most of the state of the art software has been implemented using C.

Why to use C?

C was initially used for system development work, in particular the Programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Print Spoolers
- Network Drivers
- Modern Programs
- Data Bases
- Language Interpreters
- Utilities

C Program File

All the C Programs are written into text files with extension ".c" for example ***hello.c***. You can use "vi" editor to write your C Program into a file. This tutorial assumes that you know how to edit a text file and how to write Programming instructions inside a Program file.

C Compilers

When you write any Program in C language then to run that Program you need to compile that Program using a C Compiler which converts your Program into a language understandable by a computer. This is called machine language (i.e. binary format). So before proceeding, make sure you have C Compiler available at your computer. It comes along with all flavors of UNIX and Linux. If you are working over UNIX or Linux then you can type *gcc -v* or *cc -v* and check the result. You can ask your system administrator or you can take help from anyone to identify an available C Compiler at your computer. If you don't have C compiler installed at your computer then you can use below given link to download a GNU C Compiler and use it.

Objective and Relevance

- To review basic computer systems concepts
- To be able to understand the different computing environments and their components
- To review the history of computer languages
- To be able to list and describe the classifications of computer languages
- To understand the steps in the development of a computer Program To review the system development life cycle
- To write Programs in C to solve the problems.
- To implement linear data structures such as lists, stacks, queues.
- To implement simple searching and sorting methods.

Outcomes

- This subject contributes to having students practice their writing skills with project document and report writing.
- This subject contributes to developing student critical thinking through tutorial and lab exercises on solving problems. They will also practice more in written assignments, Programming exercises, and project.
- This subject contributes to team work with group-based project for students to practice team spirit.

Code of Conduct

- Students should report to the labs concerned as per the timetable.
- Students who turn up late to the labs will in no case be permitted to perform the experiment scheduled for the day.
- After completion of the experiment, certification of the staff in-charge concerned in the observation book is necessary.
- Students should bring a notebook of about 100 pages and should enter the readings/observations/results into the notebook while performing the experiment.
- The record of observations along with the detailed experimental procedure of the experiment performed in the immediate previous session should be submitted and certified by the staff member in-charge.
- Not more than three students in a group are permitted to perform the experiment on a set up.
- The group-wise division made in the beginning should be adhered to, and no mix up of student among different groups will be permitted later.
- The components required pertaining to the experiment should be collected from Lab – in - charge after duly filling in the requisition form.
- When the experiment is completed, students should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.
- Any damage of the equipment or burnout of components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.
- Students should be present in the labs for the total scheduled duration.
- Students are expected to prepare thoroughly to perform the experiment before coming to Laboratory.
- Procedure sheets/data sheets provided to the students' groups should be maintained neatly and are to be returned after the experiment.

List of Experiments Syllabus as per JNTUA

Week – 1

a. Assemble and disassemble parts of a Computer.

Week – 2

a. Design a C program which reverses the number.
b. Design a C program which finds the second maximum number among the given list of numbers.

Week – 3

a. Construct a program which finds the kth smallest number among the given list of numbers.
b. Design an algorithm and implement using C language the following exchanges

$$\begin{tikzcd} a \arrow[rrr, bend right, dash] & b \arrow[l] & c \arrow[l] & d \arrow[l] \end{tikzcd}$$

Week – 4

a. Develop a C Program which counts the number of positive and negative numbers separately and also compute the sum of them.
b. Implement the C program which computes the sum of the first n terms of the series Sum = 1 – 3 + 5 -7 + 9

Week – 5

a. Design a C program which determines the numbers whose factorial values are between 5000 and 32565.
b. Design an algorithm and implement using a C program which finds the sum of the infinite series $1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-......$

Week – 6

a. Design a C program to print the sequence of numbers in which each number is the sum of the three most recent predecessors. Assume first three numbers as 0, 1, and 1.

Week – 7

a. Implement a C program which converts a hexadecimal, octal and binary number to decimal number and vice versa.

Week – 8

a. Develop an algorithm which computes the all the factors between 1 to 100 for a given number and implement it using C.
b. Construct an algorithm which computes the sum of the factorials of numbers between m and n.

Week – 9

a. Design a C program which reverses the elements of the array.
b. Given a list of n numbers, Design an algorithm which prints the number of stars equivalent to the value of the number. The starts for each number should be printed horizontally.

Week – 10

a. Implement the sorting algorithms
  i. Insertion sort
  ii. Exchange sort

Week – 11

a. Implement the sorting algorithms
  i. Selection sort
  ii. Partitioning sort.

Week – 12

a. Illustrate the use of auto, static, register and external variables.

Week – 13

a. Design algorithm and implement the operations creation, insertion, deletion, traversing on a singly linked list.

Week – 14

a. Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.

Week – 15

a. Design a C program which sorts the strings using array of pointers.

References:
1. B. Govindarajulu, “IBM PC and Clones Hardware Trouble shooting and Maintenance”, Tata McGraw-Hill, 2nd edition, 2002.
2. R.G. Dromey, “How to Solve it by Computer”. 2014, Pearson.

Syllabus Analysis:

S.No	Name of the Experiment	Unit No	Text /Reference Books
1	**a.** Assemble and disassemble parts of a Computer.	-	Text Book 1
2	**a.** Design a C program which reverses the number. **b.** Design a C program which finds the second maximum number among the given list of numbers.	2	Text Book 2
3	**a.** Construct a program which finds the k^{th} smallest number among the given list of numbers. **b.** Design an algorithm and implement using C language the following exchanges $\begin{tikzcd} a \arrow[rrr, bend right] & b \arrow[l] & c \arrow[l] & d \arrow[l] \end{tikzcd}$	2	Text Book 2
4	**a.** Develop a C Program which counts the number of positive and negative numbers separately and also compute the sum of them. **b.** Implement the C program which computes the sum of the first n terms of the series Sum = 1 – 3 + 5 -7 + 9	2	Text Book 2
5	**a.** Design a C program which determines the numbers whose factorial values are between 5000 and 32565. **b.** Design an algorithm and implement using a C program which finds the sum of the infinite series $1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-......$	2	Text Book 2
6	**a.** Design a C program to print the sequence of numbers in which each number is the sum of the three most recent predecessors. Assume first three numbers as 0, 1, and 1.	2	Text Book 2
7	**a.** Implement a C program which converts a hexadecimal, octal and binary number to decimal number and vice versa.	2	Text Book 2
8	**a.** Develop an algorithm which computes the all the factors between 1 to 100 for a given number and implement it using C. **b.** Construct an algorithm which computes the sum of the factorials of numbers between m and n.	3	Text Book 2
9	**a.** Design a C program which reverses the elements of the array. **b.** Given a list of n numbers, Design an algorithm which prints the number of stars	4	Text Book 2

	equivalent to the value of the number. The starts for each number should be printed horizontally.		
10	**a.** Implement the sorting algorithms **i.** Insertion sort **ii.** Exchange sort	5	Text Book 2
11	**a.** Implement the sorting algorithms **i.** Selection sort. **ii.** Partitioning sort.	5	Text Book 2
12	**a.** Illustrate the use of auto, static, register and external variables.		
13	**a.** Design algorithm and implement the operations creation, insertion, deletion, traversing on a singly linked list.		
14	**a.** Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.		
15	**a.** Design a C program which sorts the strings using array of pointers.		

Session Plan:

S.No	Week No	Unit No	Activity	Remarks
1	1	-	**a.** Assemble and disassemble parts of a Computer.	JNTUA
2	2	2	**a.** Design a C program which reverses the number. **b.** Design a C program which finds the second maximum number among the given list of numbers.	JNTUA
3	3	2	**a.** Construct a program which finds the kth smallest number among the given list of numbers. **b.** Design an algorithm and implement using C language the following exchanges $\begin{tikzcd} a & b \arrow[l] & c \arrow[l] & d \arrow[l] \arrow[lll, bend left] \end{tikzcd}$	JNTUA
4	4	2	**a.** Develop a C Program which counts the number of positive and negative numbers separately and also compute the sum of them. **b.** Implement the C program which computes the sum of the first n terms of the series Sum = 1 – 3 + 5 -7 + 9	JNTUA
5	5	2	**a.** Design a C program which determines the numbers whose factorial values are between 5000 and 32565. **b.** Design an algorithm and implement using a C program which finds the sum of the infinite series $1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-......$	JNTUA
6	6	2	**a.** Design a C program to print the sequence of numbers in which each number is the sum of the three most recent predecessors. Assume first three numbers as 0, 1, and 1.	JNTUA
7	7	2	**a.** Implement a C program which converts a hexadecimal, octal and binary number to decimal number and vice versa.	JNTUA
8	8	3	**a.** Develop an algorithm which computes the all the factors between 1 to 100 for a given number and implement it using C. **b.** Construct an algorithm which computes the sum of the factorials of numbers between m and n.	JNTUA
9	9	4	**a.** Design a C program which reverses the elements of the array. **b.** Given a list of n numbers, Design an algorithm which prints the number of stars	JNTUA

			equivalent to the value of the number. The starts for each number should be printed horizontally.	
10	10	5	**a.** Implement the sorting algorithms **i.** Insertion sort **ii.** Exchange sort	JNTUA
11	11	5	**a.** Implement the sorting algorithms **i.** Selection sort. **ii.** Partitioning sort.	JNTUA
12	12		**a.** Illustrate the use of auto, static, register and external variables.	JNTUA
13	13		**a.** Design algorithm and implement the operations creation, insertion, deletion, traversing on a singly linked list.	JNTUA
14	14		**a.** Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.	JNTUA
15	15		**a.** Design a C program which sorts the strings using array of pointers.	JNTUA

Equipment Required:

Hardware:

No. of System	:	60(IBM)
Processor	:	PIV™ 1.67 GH_z
RAM	:	512 MB
Hard Disk	:	40 GB
Mouse	:	Optical Mouse
Network Interface card	:	Present

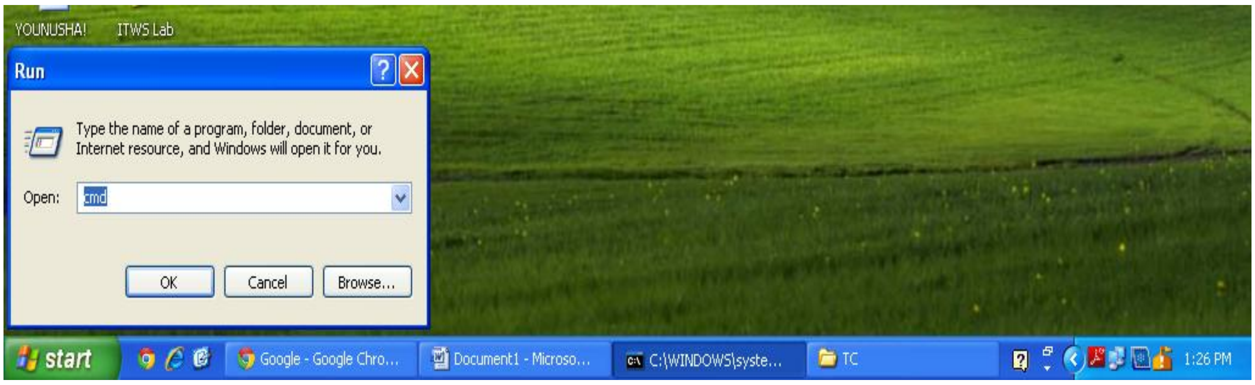
Software:

Operating System	:	Window XP
Application Software	:	Turbo C.

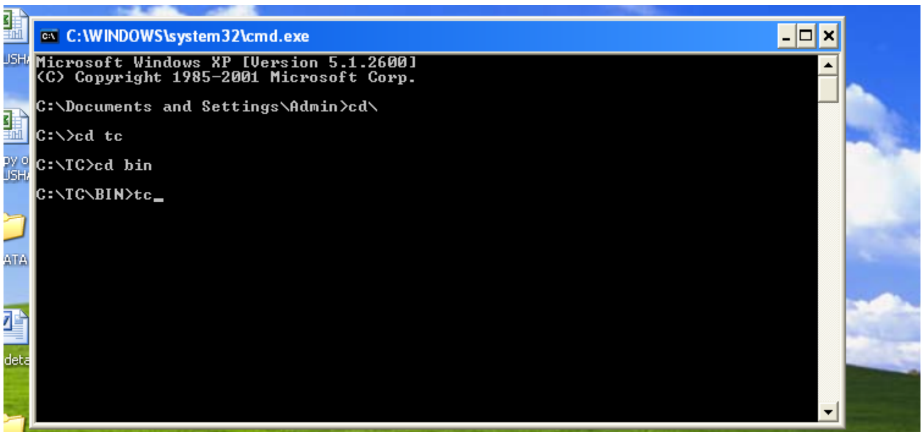
Experimental Write Up:

Introduction

How to Open Application

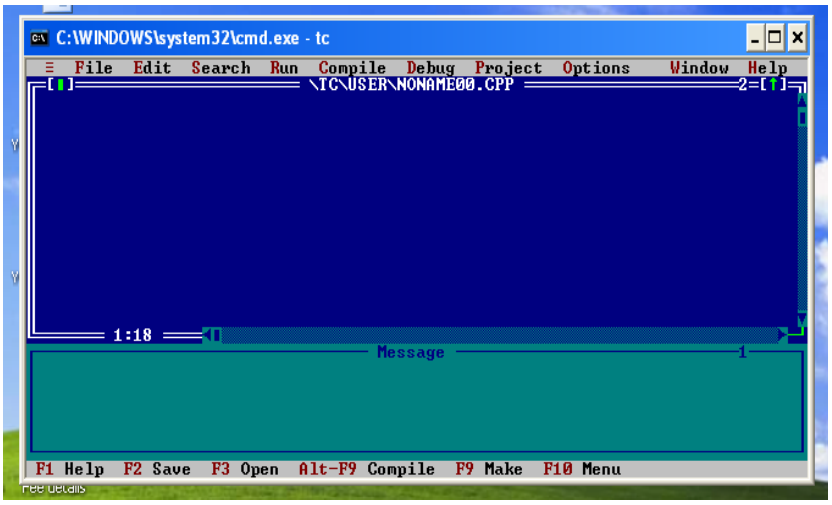

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open: cmd
OK
Cancel
Browse...
start
1:26 PM

Step 1: Open Command Prompt Window



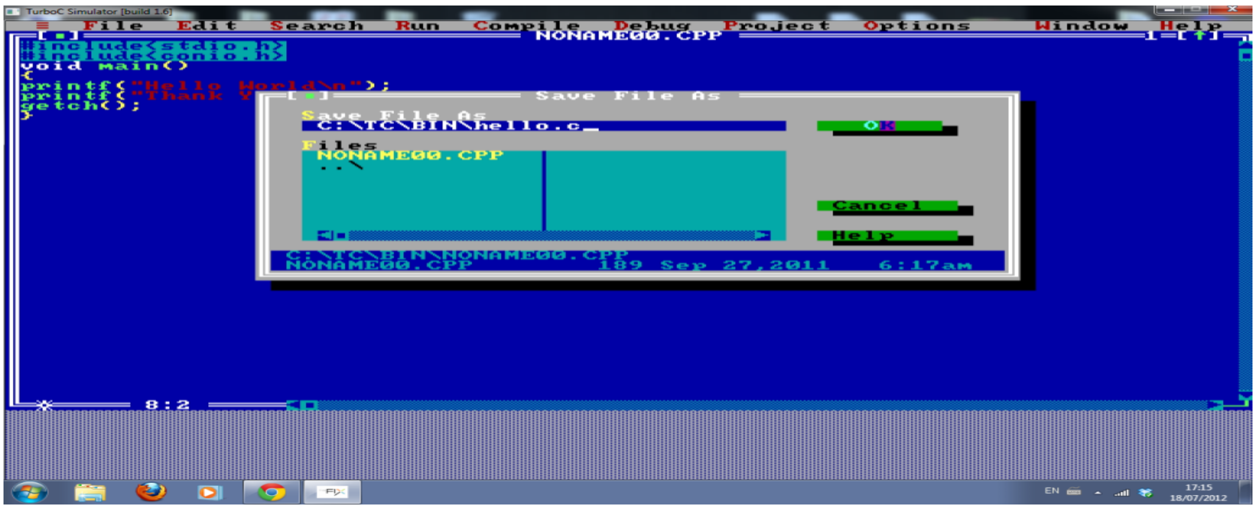
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Admin>cd\

C:\>cd tc

C:\TC>cd bin

C:\TC\BIN>tc_
```

Step 2: Open TC with Command Prompt Window


C:\WINDOWS\system32\cmd.exe - tc
≡ File Edit Search Run Compile Debug Project Options Window Help
\TC\USER\NONAME00.CPP
1:18
Message
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
void main()
{
printf ("Hello World\n");
printf ("Thank You");
getch ();
 }
```

I hope that's enough for a basic explanation of the Program. If you still have doubts please ask through comments. Now let's **RUN this Program using Turbo C**. Before going into the steps, you may SAVE your C Program. Select **"File"** from menu -> click-> **Save**. Name the files as ->**hello.c** or some other name with a .c extension. See the screen shot below.

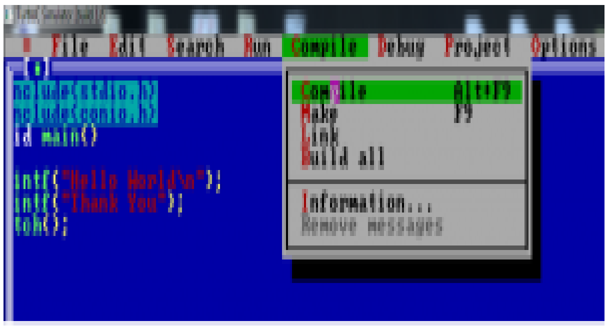

TurboC Simulator [build 1.6]
File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP
Save File As
OK
Cancel
Help
8:2

```
#include<stdio.h>
#include<conio.h>
void main()
{
printf("Hello World\n");
printf("Thank You");
getch();
}
```

Save File As
C:\TC\BIN\hello.c
Files
NONAME00.CPP
..\
C:\TC\BIN\NONAME00.CPP
NONAME00.CPP 189 Sep 27,2011 6:17am

How to Compile a C Program in Turbo C:

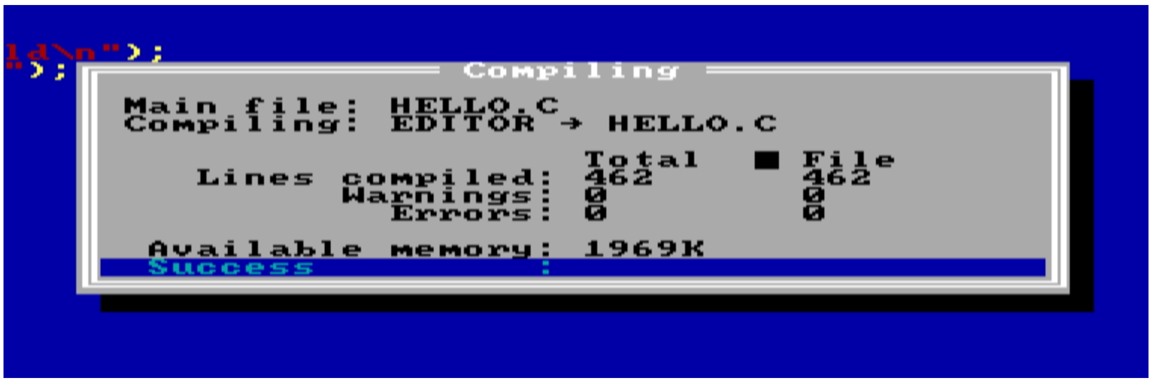
The first step is compiling. Compiling makes sure your Program is **free of syntax errors**. However compiler won't check for any logical/Algorithmic errors. There is a lot of process that happens while the compiler compiles a Program – which we will discuss later in coming articles. To do compiling - **Select -> Compile** from menu and **click-> compile**. See the image below.



```
nclude<stdio.h>
nclude<conio.h>
id main()

intf("Hello World\n");
intf("Thank You");
tch();
```

After compiling, you will see a dialog box as shown below. If the compilation is success – you will see a **"success"** message. Else you will see the number of errors. Both are shown using screen shots.

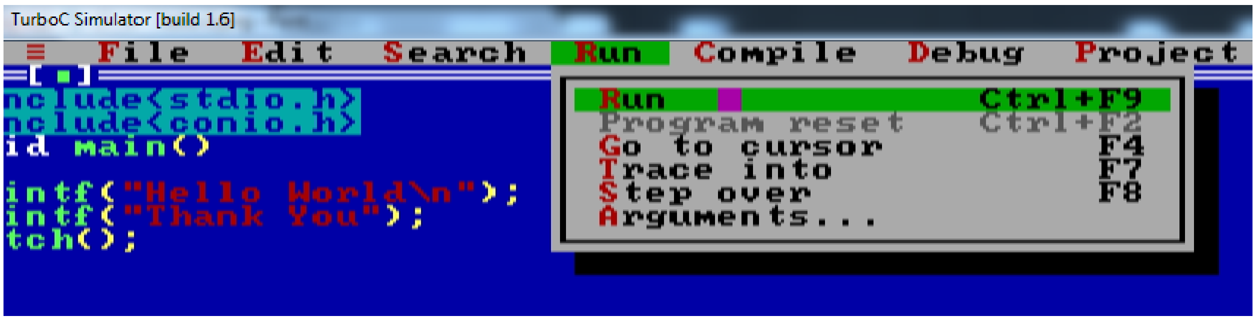
The screen shot of a "success" compilation



```
ld\n");
");
                      Compiling
 Main file: HELLO.C
 Compiling: EDITOR → HELLO.C

                      Total   File
     Lines compiled:  462     462
           Warnings:  0       0
             Errors:  0       0

  Available memory:   1969K
  Success         :
```

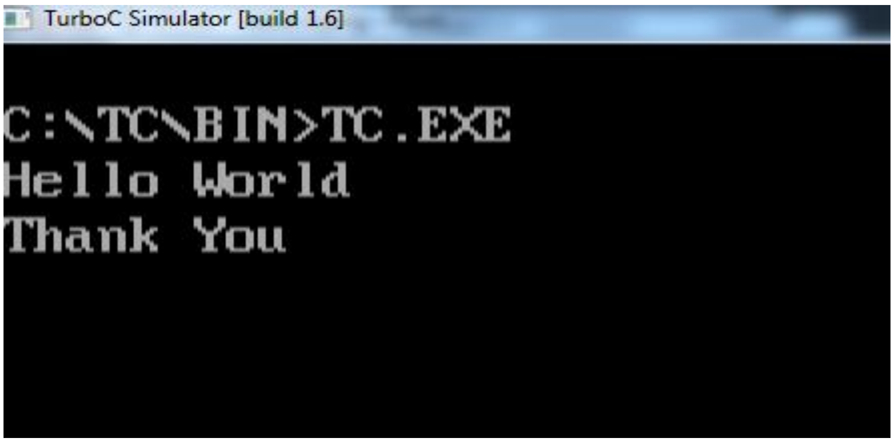
How to RUN a C Program in Turbo C compiler?

To RUN the Program – you may **select ->Run** from menu and **click -> Run** (as shown in the image below).



```
nclude<stdio.h>
nclude<conio.h>
id main()

intf("Hello World\n");
intf("Thank You");
tch();
```

Now you will see the output screen as shown in the screen shot below.



```
C:\TC\BIN>TC.EXE
Hello World
Thank You
```

Week - 1:

Aim: Assemble and disassemble parts of a Computer.

Requirements:

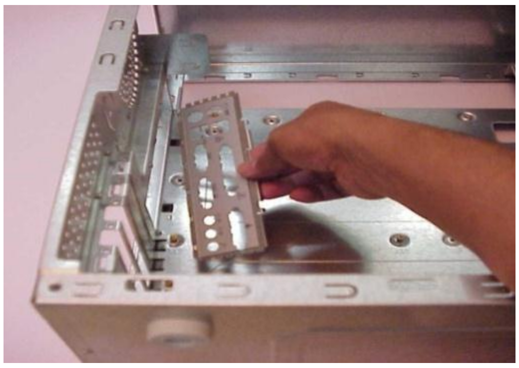
1. CPU(Processor)
2. Mother Board
3. Floppy Disk Drive
4. Hard Disk Drive
5. CD or DVD ROM
6. Cabinet
7. Speakers
8. Key Board
9. Mouse
10. Monitor
11. RAM(SD or DDR)
12. Bus Cables
13. Power Cables
14. SMPS
15. Screw Driver
16. Screws
17. Printer etc…

Procedure:

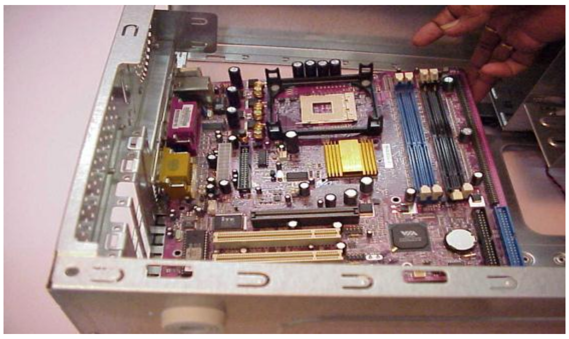
Mother Board Installation:

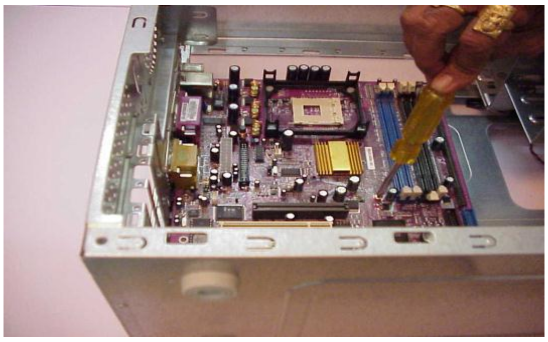
a. Open the cabinet on either side.

b. The back side of the cabinet has readymade provision for the installation of the I/O shields. An I/O shield is used for connecting the input and output devices through it.



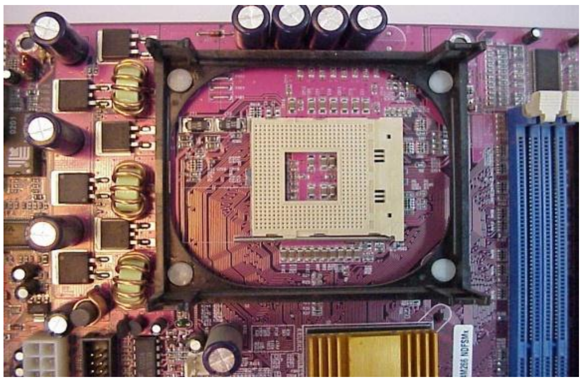
c. Check whether the mother board is placed in such a way that the I/O ports of the motherboard correctly fit in the I/O shields. Ensure all the specified screws for the motherboard are fixed and intact.



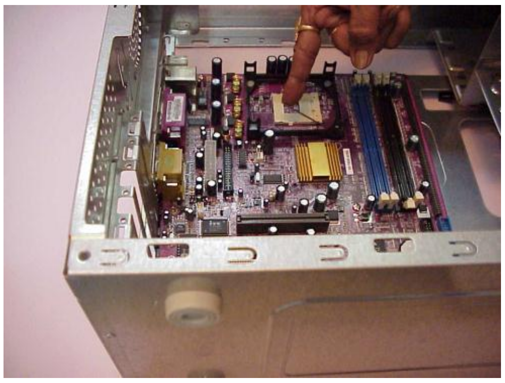


CPU Installations:

1. CPU is one of the most dedicated components of the computer. The CPU pins have to be clearly studied before fixing into the relevant processor space on the motherboard. After the CPU is rightly placed in its position the lever is to be locked.

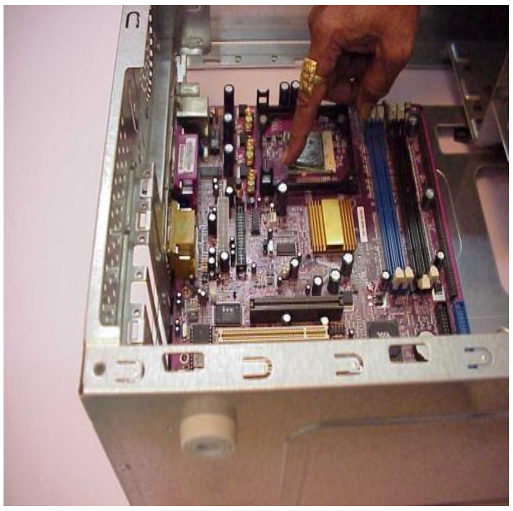


2. As a part of the CPU installation, before the CPU is fixed in the right position a lever is provided, which needs to be unlocked. This lever is perpendicular to the motherboard.



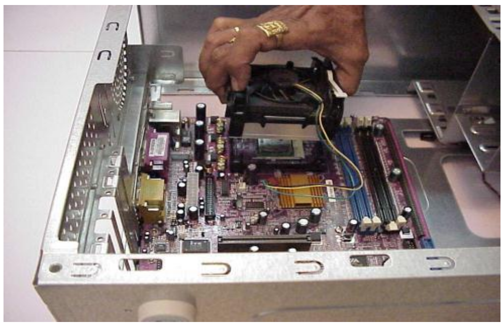
3. The CPU, which is a square shaped electronic component, comes with pins below it. One should find for an indication on one of the corners of the CPU on both sides. This arrow mark is also found on the motherboard which guides for the fixation of the CPU. Once match of the pins verses motherboard slot gently push the CPU.

4. After the CPU is rightly placed in its position, the lever is to be locked.

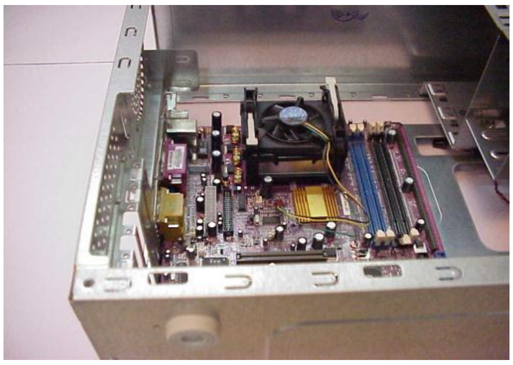


CPU heat sink fan installation:

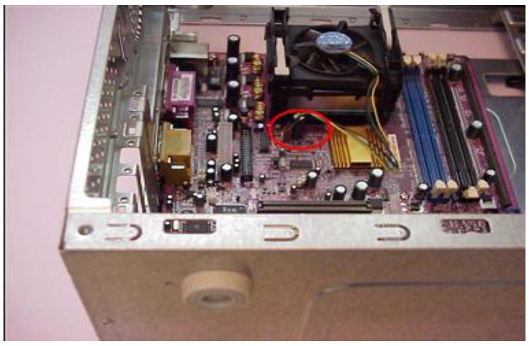
5. The CPU heat sink fan is to be carefully plugged on to the CPU by pushing down the metal plastic clips.



6. The metal/plastic clips provided with heat sink fan should fix on to the CPU socket and have to be locked.

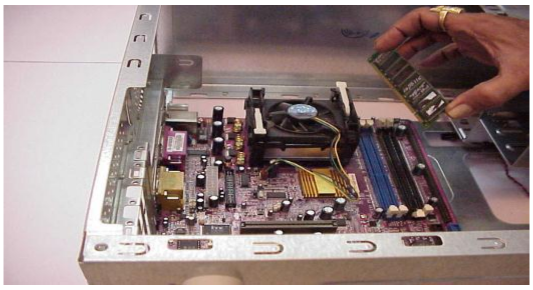


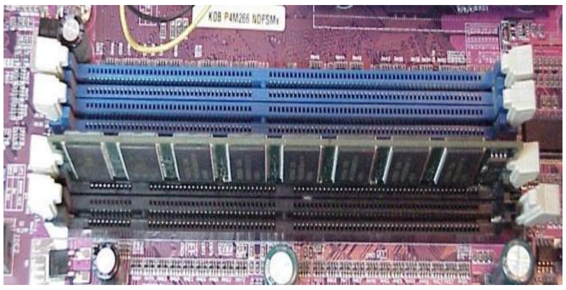
7. Once the CPU het sink fan is fixed and locked, it should be connected to the Power supply available on the mother board through the power connector.



RAM Installation:

8. Next is installing the RAM. Insert the RAM into an available expansion socket. Note how the RAM is keyed to the socket. This ensures the RAM can be plugged into the socket one way only. Finally press the RAM firmly into position, making certain the Ram is completely seated in the socket.

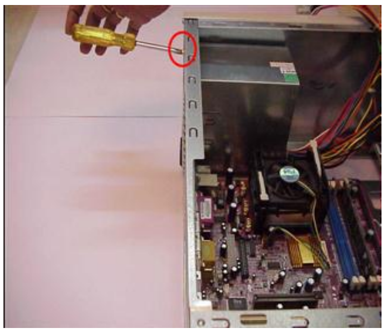




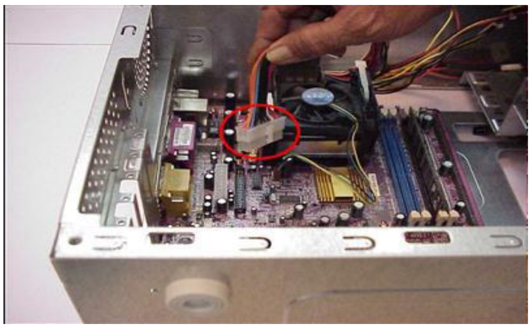
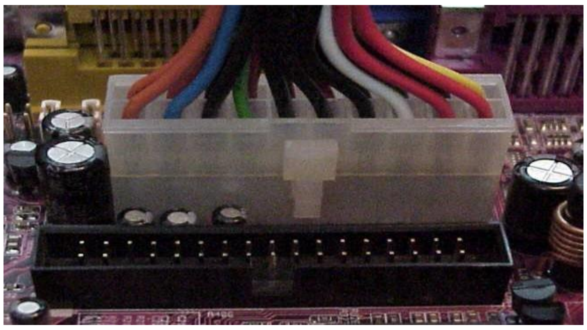
SMPS Installations:

9. Next is installing the SMPS. This is an electronic power supply unit that provides and regulates the power supply to all components of a computer system. As shown in the diagram the SMPS needs to install into cabinet at the place provided for it.

10. After placing the SMPS into the relevant provider space fix the outer screws to it intact.

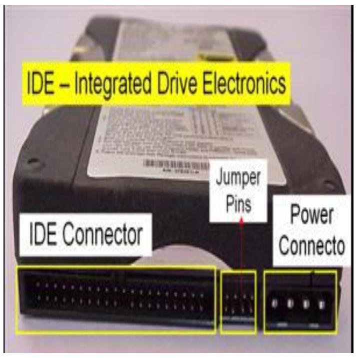


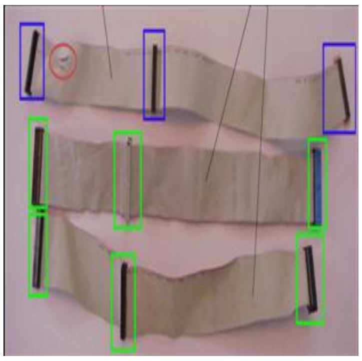
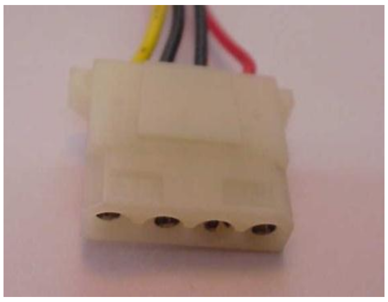
11. Next installing the ATX power connector. It is a 20/24-pin power connector. This is the primary power supply to the mother board.

Hard Disk Drive Installation:

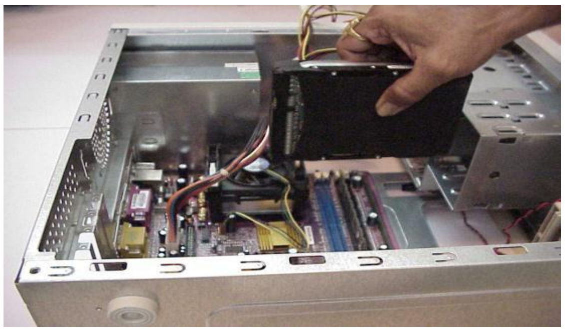
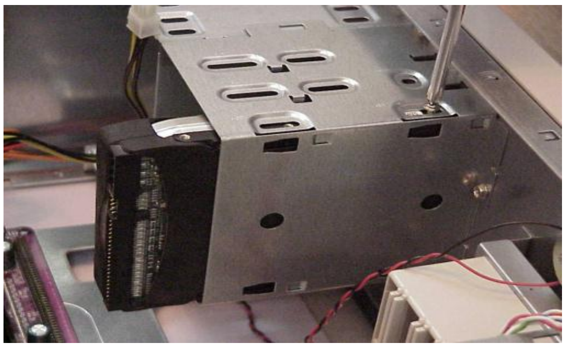
12. Installing the Hard Disk Drive (HDD) is clearly understood in the following steps. First see the rare of the HDD. It consists of the 3 types of pins. One left side the HDD has multiple pins termed as the IDE connector. In the middle is the jumper setting pins for the HDD. On the extreme right side is the power connector pins. Every device except FDD (floppy Disk Drive) uses this type of power connector. And HDD and CDD (Compact Disk Drive) connected by this type of IDE cable.



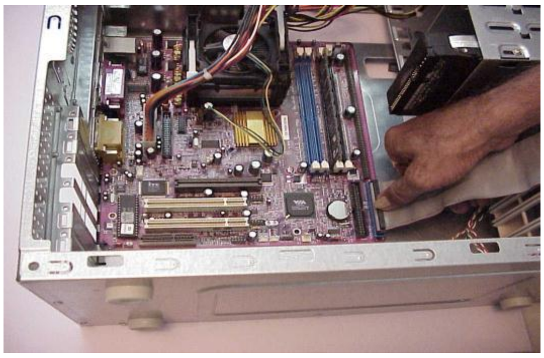
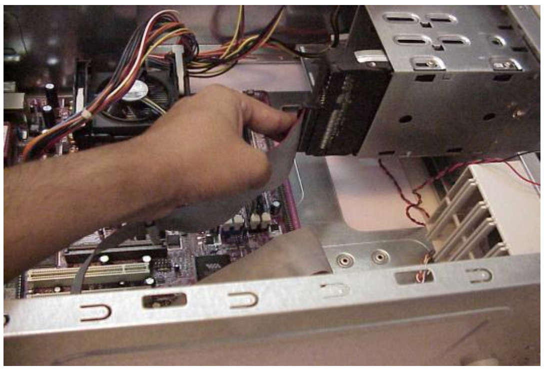



Power Connector

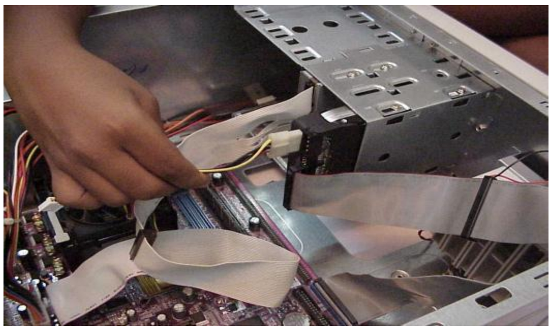
13. Mount the HDD into mounting slot meant for the HDD with the rear end facing and secure the inner screws intact.

14. Connect the IDE cable to the HDD as well as the mother board as shown in the figure.

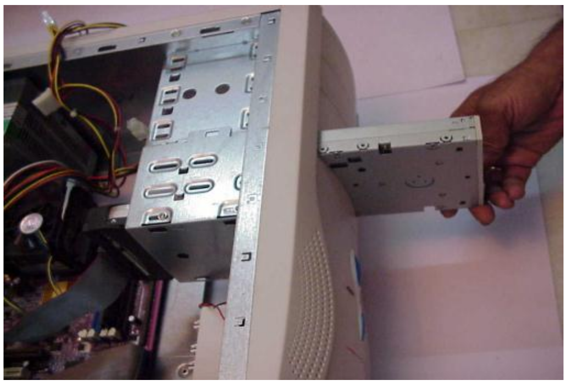
15. Remember for all the power connectors to be plugged in, one needs to align the Red line on the cable to Pin-1 of the IDE port. Hence connect the power cable to the HDD rare end by gently pushing the connector.



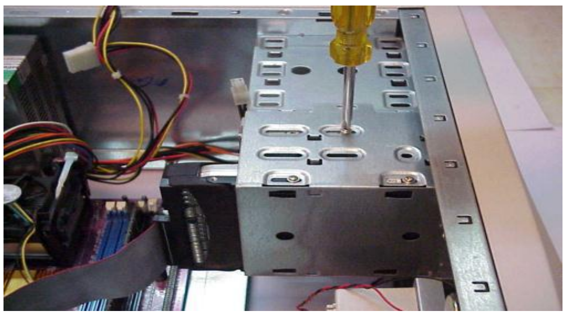
Floppy Disk Drive Installation:

16. Installation of a Floppy Disk Drive (FDD) is very similar to the HDD installation. We need to identify the relevant pins for the motherboard and power supply connectivity. First Step in the FDD installation is mounting of the FDD into the FDD mounting slot by removing the cover of front side of the cabinet as shown in the figure below.

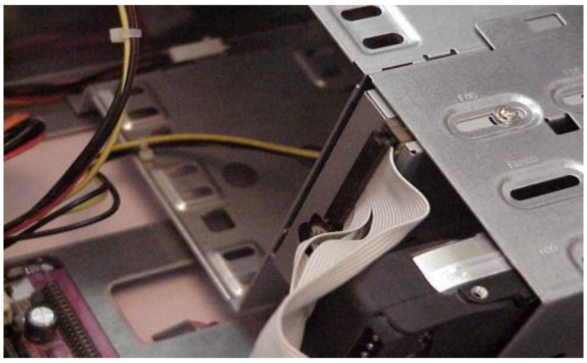
17. Push the FDD case into opened of the cabinet curtaining of the FDD



18. Secure FDD with inner screws.



19. Connect the one end of cable to mother board and other top end to FDD and also connect the power cable.

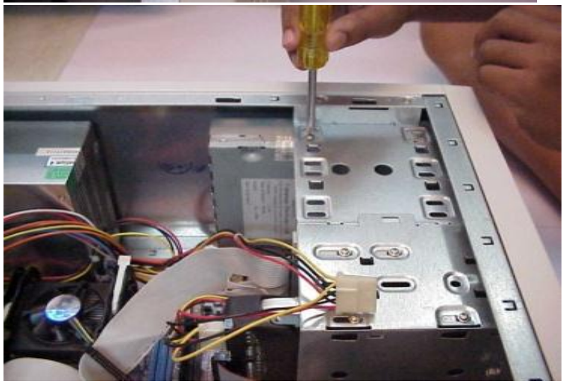


CD ROM Installation:

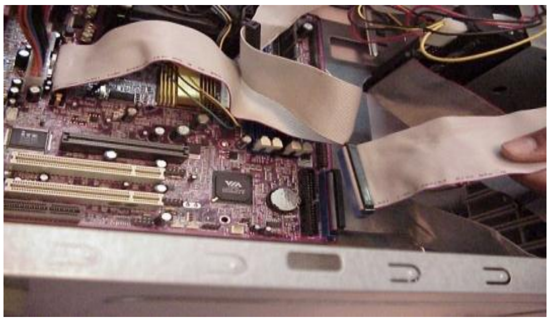
20. Next installing the CD-ROM. Remove the cover of front side of the cabinet curtaining of the CD-ROM.

21. Push CD-ROM case into opened space.

22. Secure CD-ROM with inner screws.

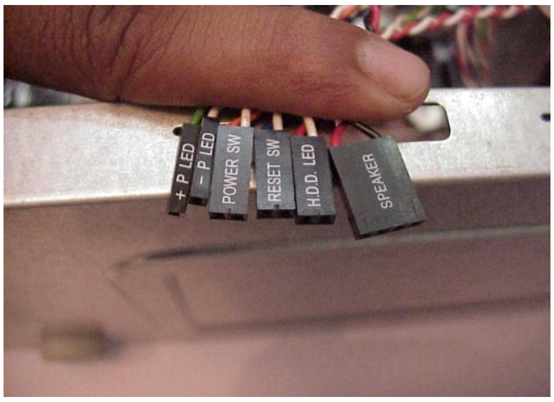


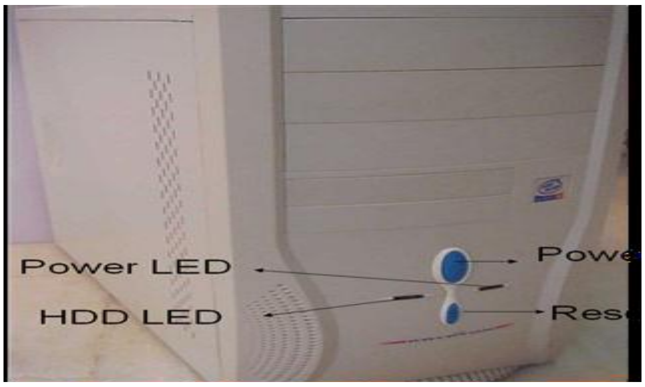
23. Connect the one end of cable to motherboard and another end to CD-ROM and connect the power connector.



Switches and LEDs Connections:

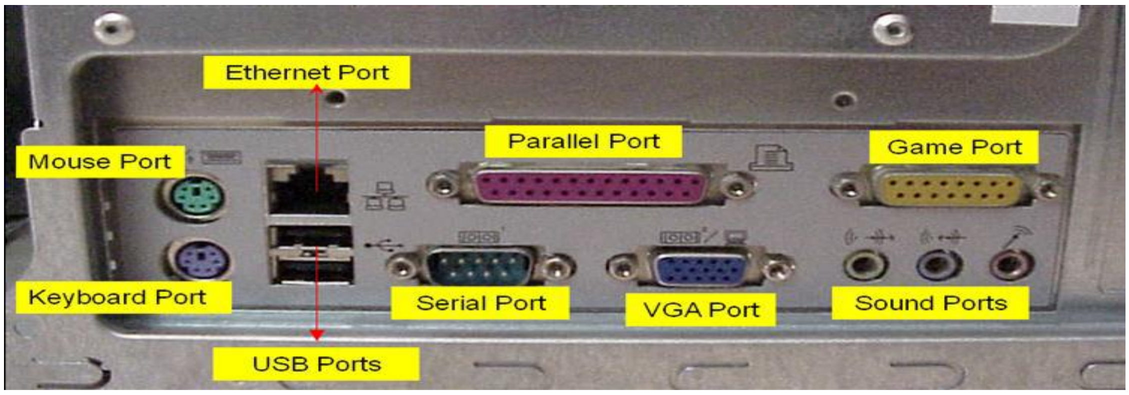
24. Installing the Switches and LEDs of front side of the cabinet. Please refer to your mother board manual to locate where the connectors are. Different mother boards place the connectors in different locations. The connectors for the switches and LEDs are normally grouped together. They should look similar to the figure given below.





IO Devices Installations:

25. Finally connect all peripheral devices like mouse, key-board, monitor, etc, to the I/O ports shown in the figure below.



a) **Keyboard:** Keyboard has round shape connectors. The male connector appears at the edge of the keyboard's cable and the female connector appears at the back side of the system unit. We are using the 6 pins round keyboard connector.

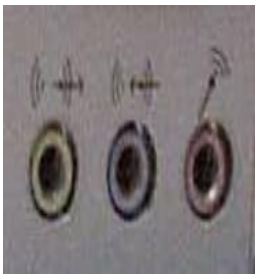
b) **Mouse:** The mouse connector is same as the keyboard connector. The male connector appears at the edge of mouse cable and female connector appears at the backside of the system. It is also having 6 pins to connect the mouse.

c) **Monitor:** The monitor of computer has " D "̛ shape connectors. The male Monitor connector has 15 pins and it appears at the edge of monitor"s cable. The female monitor connector appears at the back of the system unit.

d) Printer:

Printer connector is the oldest connector of a computer. The male printer connector has 25 pins and it appears at the edge of the printer cable and the backside of the system unit.

e) Audio / Speaker: For audio effect we are using speakers. The audio male connector have single thick pin and each male connector of individual speaker is distinguish with separate color. The male connectors appear at the edge of the speaker cables. The female audio connectors appear in same color at the back side of the system unit. The female audio connectors have some special symbols i.e.



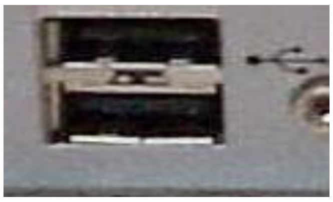
The first symbol displays "line-out" - it sends the output to speakers.

The second symbol displays "line-in"- it takes the input from speakers.

The third symbol displays "Mic-in" - it takes the input from microphone.

f) Ethernet / Networking: The Ethernet connectors are used when two or more than two computers need to be linked with other over a computer network like LAN (local area network). The shape of male Ethernet connector is quite similar to male modem connector except it is more flat. The female Ethernet connector appears at the back of the system unit.

g) USB: USB (universal serial bus) is the latest and most popular connector. Using USB connectors, we can connect so many different devices to our computer. Any device equipped with USB has slim male connector with slim metal coating appearing at the end of the devices cable. For connecting the device, a female USB connector is provided at the back of the system unit. We can identify the USB connector with this symbol.



Result: personal computer successfully assembled.

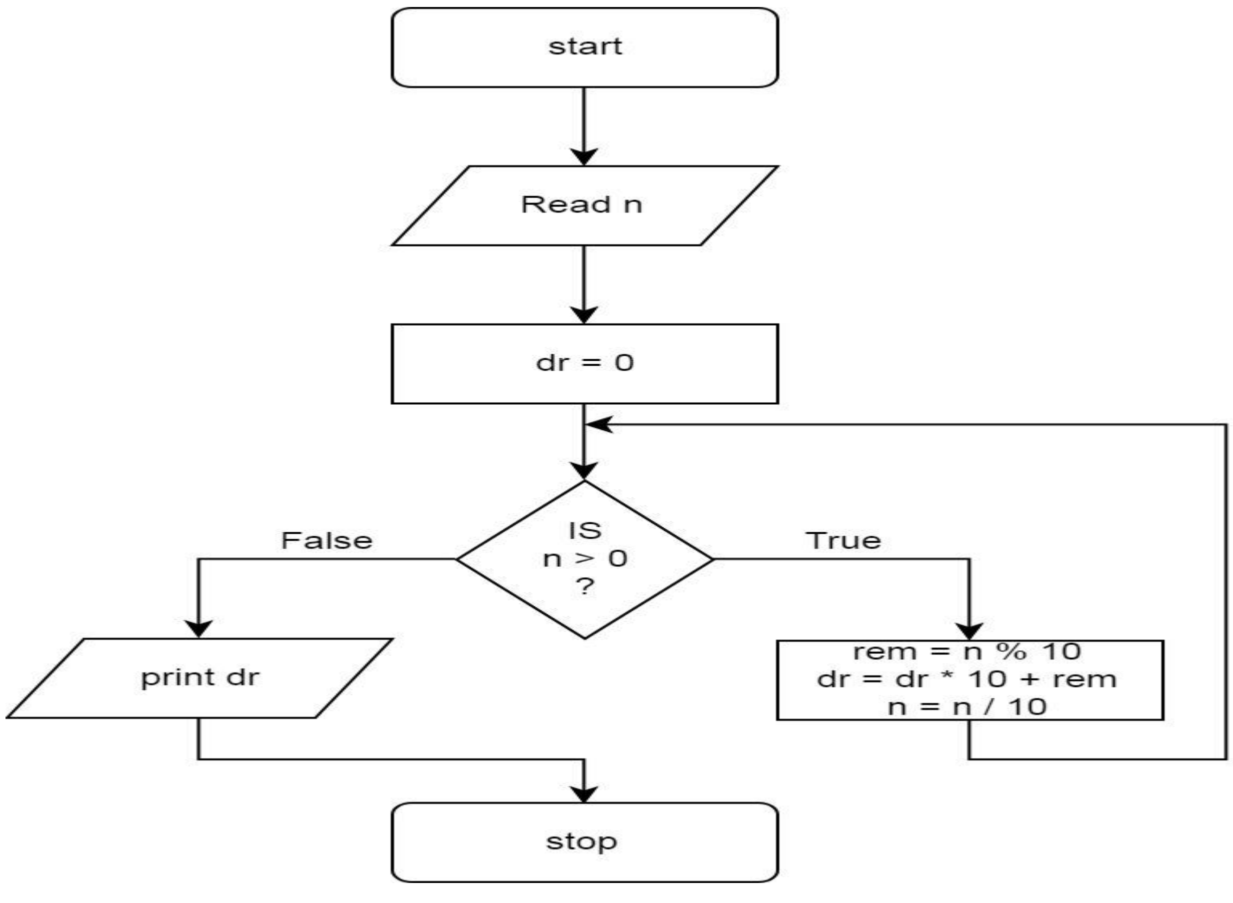
Week – 2:

Aim: Design a C program which reverses the number.

Algorithm:

```
start
print "Enter any number for reverse"
input num
rev ← 0
while num > 0 repeatedly do
begin
     rem = num % 10
     dreverse = dreverse * 10 + rem
     num = num / 10
end
print reverse
stop
```

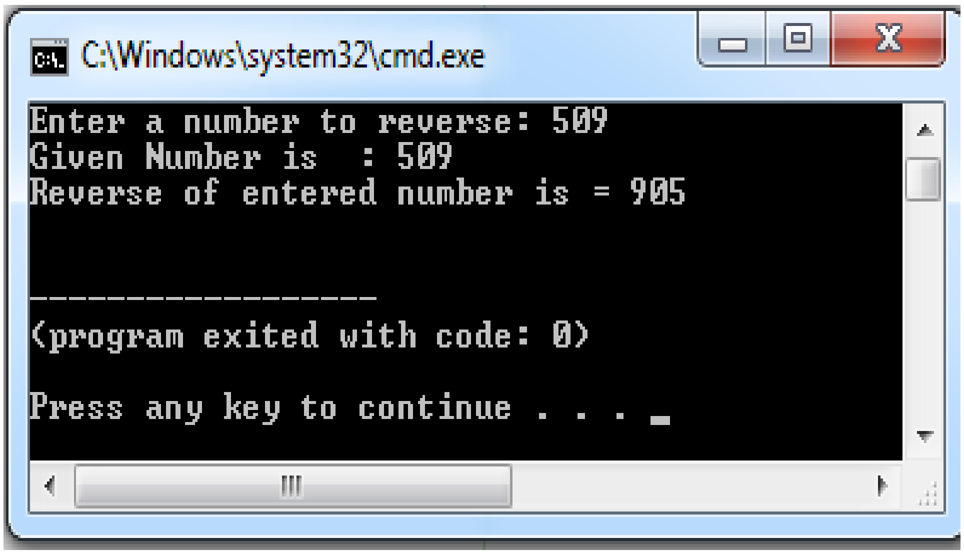
Flowchart:



Program:

```c
#include <stdio.h>

int main()
{
     int n, dreverse = 0, remainder;
     printf("Enter a number to reverse: ");
     scanf("%d", &n);
     printf("Given Number is  : %d\n",n);
     while (n != 0)
     {
          remainder = n % 10;
          dreverse  = (dreverse *10) + remainder;
          n         = n/10;
     }
     printf("Reverse of entered number is = %d\n", dreverse);
}
```

Output:



```
Enter a number to reverse: 509
Given Number is  : 509
Reverse of entered number is = 905


------------------
(program exited with code: 0)

Press any key to continue . . .
```

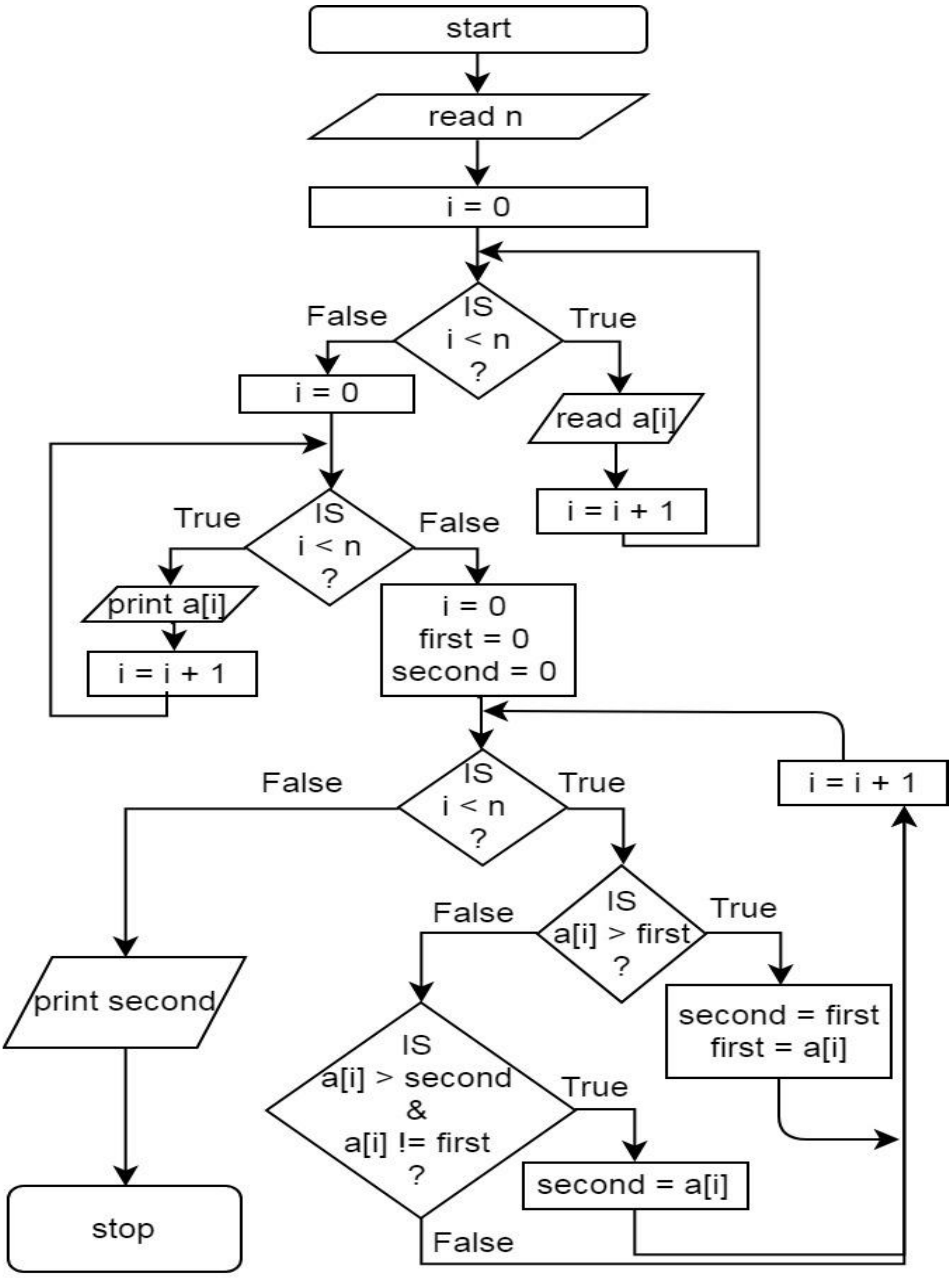
Result: The above program successfully executed.

Aim: Design a C program which finds the second maximum number among the given list of numbers.

Algorithm:

```
start
print “enter the no. Of elements in the array”
input n
i ← 0
l1:  print “enter array element”
     input list[i]
     i ← i + 1
     if i<n then
          goto l1
print “array elements”
i ← 0
l2:  print list[i]
     i ← i + 1
     if i<n then
          goto l2
first ← 0
second ← 0
i ← 0
l3:  if list[i] > first then
          second ← first
          first ← list[i]
     else if list[i] > second AND list[i] != first then
          second ← list[i]
     i ← i + 1
     if i < n then
          goto l3
if second = 0 then
     print “There is no second largest element”
else
     print list[k-1]
stop
```

Flowchart:

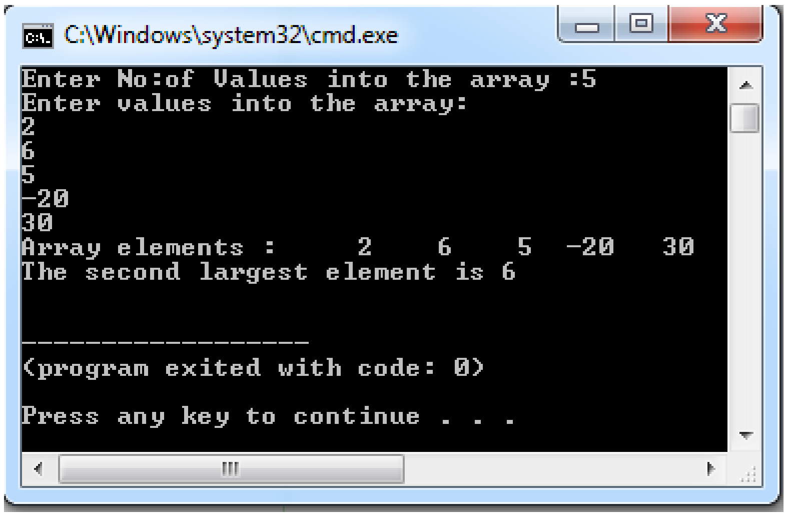

start
read n
i = 0
IS i < n ?
False
True
read a[i]
i = i + 1
i = 0
IS i < n ?
True
False
print a[i]
i = i + 1
i = 0
first = 0
second = 0
IS i < n ?
False
True
i = i + 1
IS a[i] > first ?
False
True
second = first
first = a[i]
IS a[i] > second & a[i] != first ?
True
False
second = a[i]
print second
stop

Program:

```
#include <stdio.h>
#include <limits.h>

int main()
{
     int n, i, arr[10], first, second;
     printf("Enter No:of Values into the array :");
     scanf("%d",&n);
     printf("Enter values into the array:\n");
     for(i=0;i<n;i++)
          scanf("%d",&arr[i]);

     /* There should be atleast two elements */
     if (n < 2)
     {
          printf(" Invalid Input ");
          return 0;
     }
     printf("Array elements : ");
     for (i = 0; i < n ; i ++)
     {
          printf("%5d", arr[i]);
     }

     first = second = INT_MIN;
     for (i = 0; i < n ; i++)
     {
          /* If current element is greater than first
          then update both first and second */
          if (arr[i] > first)
          {
               second = first;
               first = arr[i];
          }

          /* If arr[i] is in between first and
          second then update second */
```

```
            else if (arr[i] > second && arr[i] != first)
                 second = arr[i];
       }

       if (second == INT_MIN)
            printf("\nThere is no second largest element\n");
       else
            printf("\nThe second largest element is %d\n", second);
       return 0;
}
```

Output:



```
Enter No:of Values into the array :5
Enter values into the array:
2
6
5
-20
30
Array elements :     2    6    5  -20   30
The second largest element is 6


------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

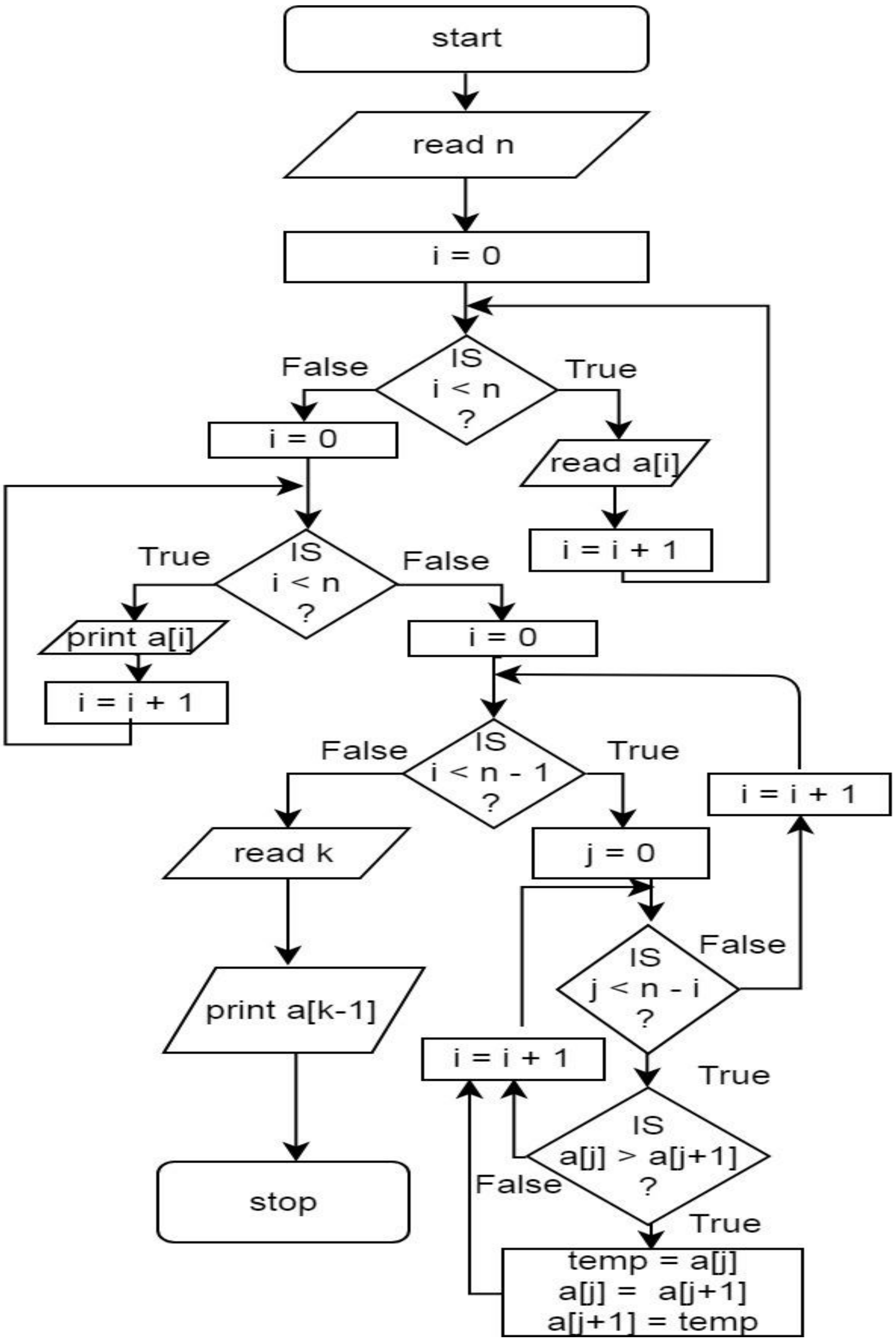
Week - 3:

Aim: Construct a program which finds the k^{th} smallest number among the given list of numbers.

Algorithm:

```
start
print “enter the no. Of elements in the array”
input n
i ← 0
l1:  print “enter array element”
     input list[i]
     i ← i + 1
     if i<n then
          goto l1
print “array elements”
i ← 0
l2:  print list[i]
     i ← i + 1
     if i<n then
          goto l2
print “Enter to find out kth smallest element”
read k
i ← 0
l3:  j ← 0
     l4:  if list[j] > list[j+1] then
               temp ← list[j]
               list[j] ← list[j+1]
               list[j+1] ← temp
          j ← j + 1
          if j < n – i - 1 then
               goto l4
     i ← i + 1
     if i < n - 1 then
          goto l3
print “kth smallest element is”
print list[k-1]
stop
```

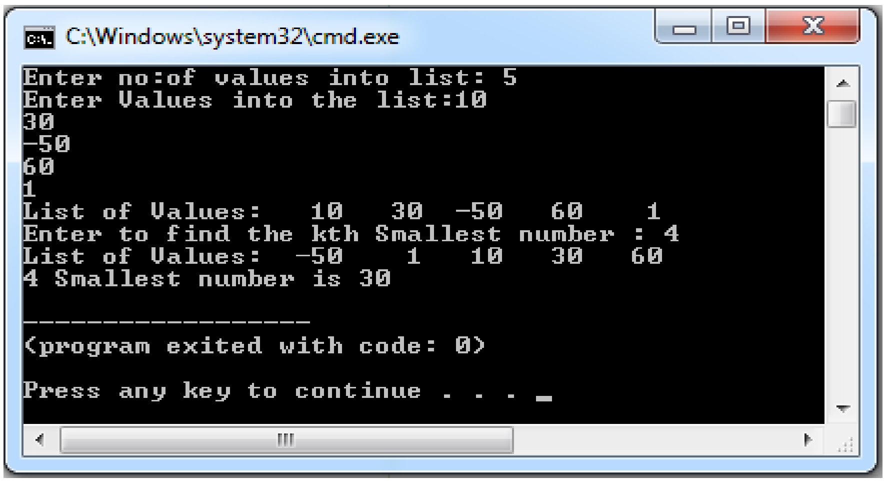
Flowchart:


start
read n
i = 0
IS i < n ?
False
True
i = 0
read a[i]
i = i + 1
IS i < n ?
True
False
print a[i]
i = i + 1
i = 0
IS i < n - 1 ?
False
True
read k
j = 0
i = i + 1
IS j < n - i ?
False
True
print a[k-1]
i = i + 1
IS a[j] > a[j+1] ?
False
True
stop
temp = a[j]
a[j] = a[j+1]
a[j+1] = temp

Program:

```
#include<stdio.h>
void main()
{
     int i, n, arr[10], k, temp, j;
     printf("Enter no:of values into list: ");
     scanf("%d", &n);
     printf("Enter Values into the list:");
     for(i = 0; i < n; i++)
          scanf("%d", &arr[i]);
     printf("List of Values:");
     for(i = 0; i < n; i++)
          printf("%5d", arr[i]);
     printf("\nEnter to find the kth Smallest number : ");
     scanf("%d", &k);
     for (i = 0 ; i < n - 1; i++)
     {
          for (j = 0 ; j < n - i - 1; j++)
          {
               if (arr[j] > arr[j+1])
               {
                    temp       = arr[j];
                    arr[j]    = arr[j+1];
                    arr[j+1] = temp;
               }
          }
     }
     printf("\n%d Smallest number is %d", k, arr[k-1]);
}
```

Output:


C:\Windows\system32\cmd.exe

```
Enter no:of values into list: 5
Enter Values into the list:10
30
-50
60
1
List of Values:   10   30  -50   60    1
Enter to find the kth Smallest number : 4
List of Values:  -50    1   10   30   60
4 Smallest number is 30

------------------
(program exited with code: 0)

Press any key to continue . . . _
```

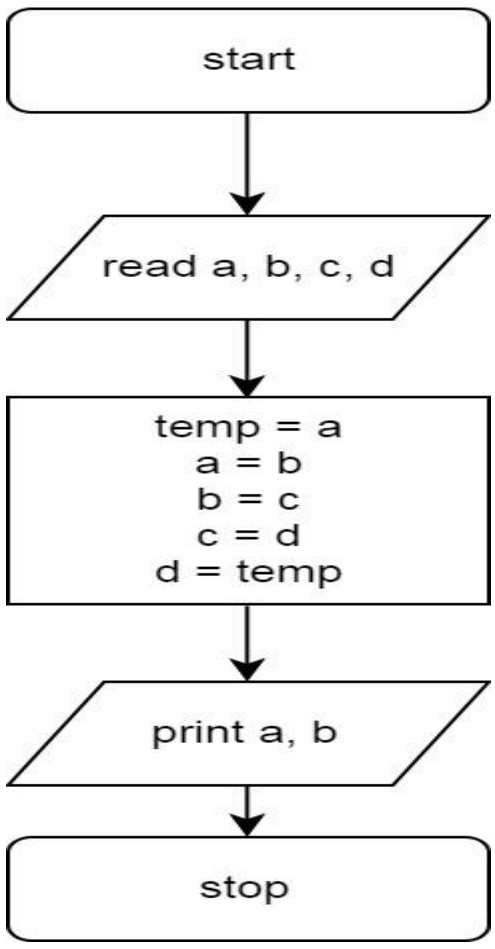
Result: The above program successfully executed.

Aim: Design an algorithm and implement using C language the following exchanges $$\begin{tikzcd} a \arrow[rrr, bend right] & b \arrow[l] & c \arrow[l] & d \arrow[l] \end{tikzcd}$$

Algorithm:

```
start
print “Enter any four values for exchanging”
input a, b, c, d
temp ← a
a ← b
b ← c
c ← d
d ← temp
print a, b, c, d
stop
```

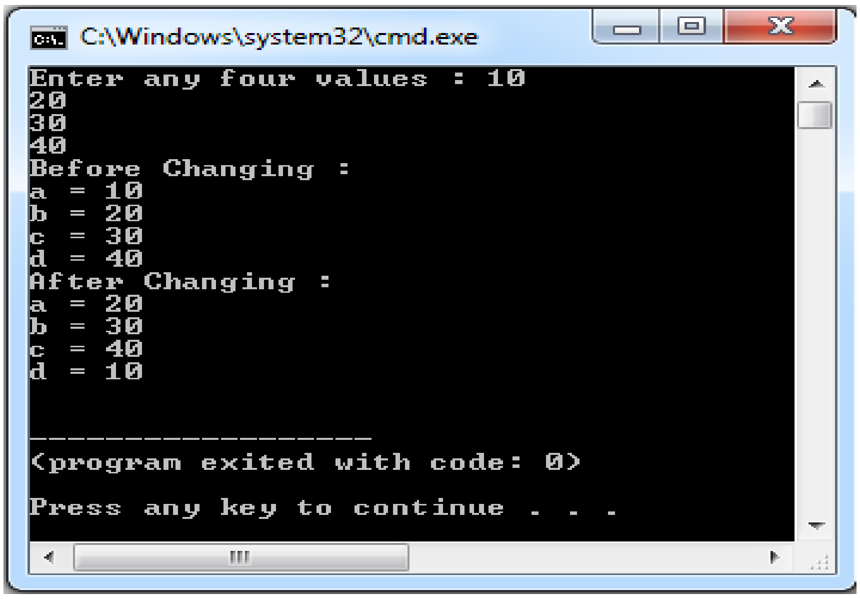
Flowchart:



Program:

```
#include<stdio.h>

int main()
{
      int a, b, c, d, temp;
      printf("Enter any four values : ");
      scanf("%d%d%d%d", &a, &b, &c, &d);
      printf("Before Changing : \n");
      printf("a = %d \nb = %d\nc = %d\nd = %d\n", a, b, c, d);
      temp = a;
      a = b;
      b = c;
      c = d;
      d = temp;
      printf("After Changing : \n");
      printf("a = %d \nb = %d\nc = %d\nd = %d\n", a, b, c, d);
}
```

Output:



```
Enter any four values : 10
20
30
40
Before Changing :
a = 10
b = 20
c = 30
d = 40
After Changing :
a = 20
b = 30
c = 40
d = 10


--------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

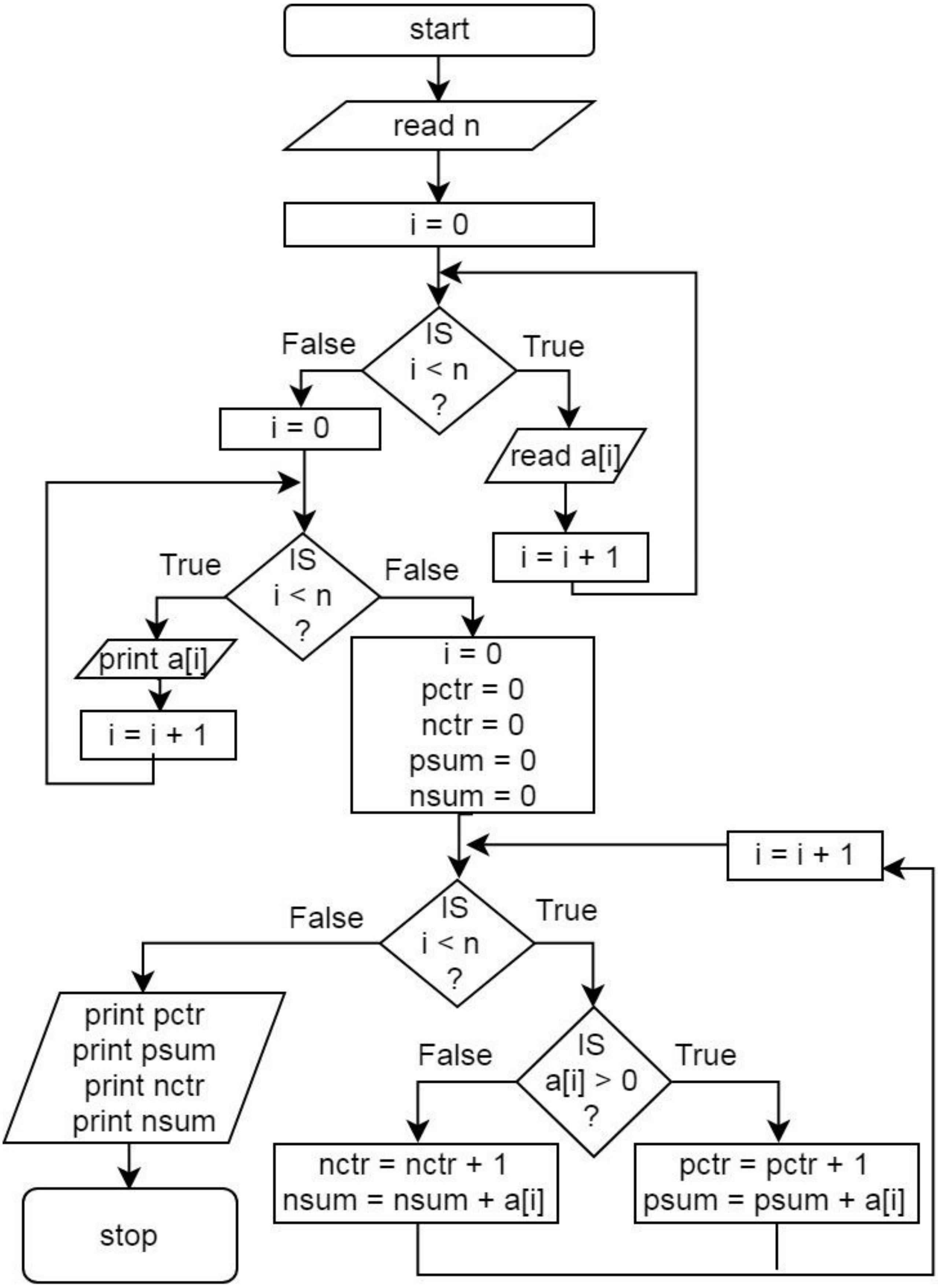
Week - 4:

Aim: Develop a C Program which counts the number of positive and negative numbers separately and also compute the sum of them.

Algorithm:

```
start
print “enter the no. Of elements in the array”
input n
i ← 0
l1:  print “enter array element”
     input list[i]
     i ← i + 1
     if i < n then goto l1
print “array elements”
i ← 0
l2:  print list[i]
     i ← i + 1
     if i<n then
          goto l2
psum ← 0
nsum ← 0
pctr ← 0
nctr ← 0
i ← 0
l3:  if list[i] > 0 then
          pctr ← pctr + 1
          psum ← psum + list[i]
     else
          nctr ← nctr + 1
          nsum ← nsum + list[i]
     if i < n then
          goto l3
print “positive count “+pctr
print “negative count “+nctr
print “positive sum “+psum
print “negative sum “+nsum
stop
```

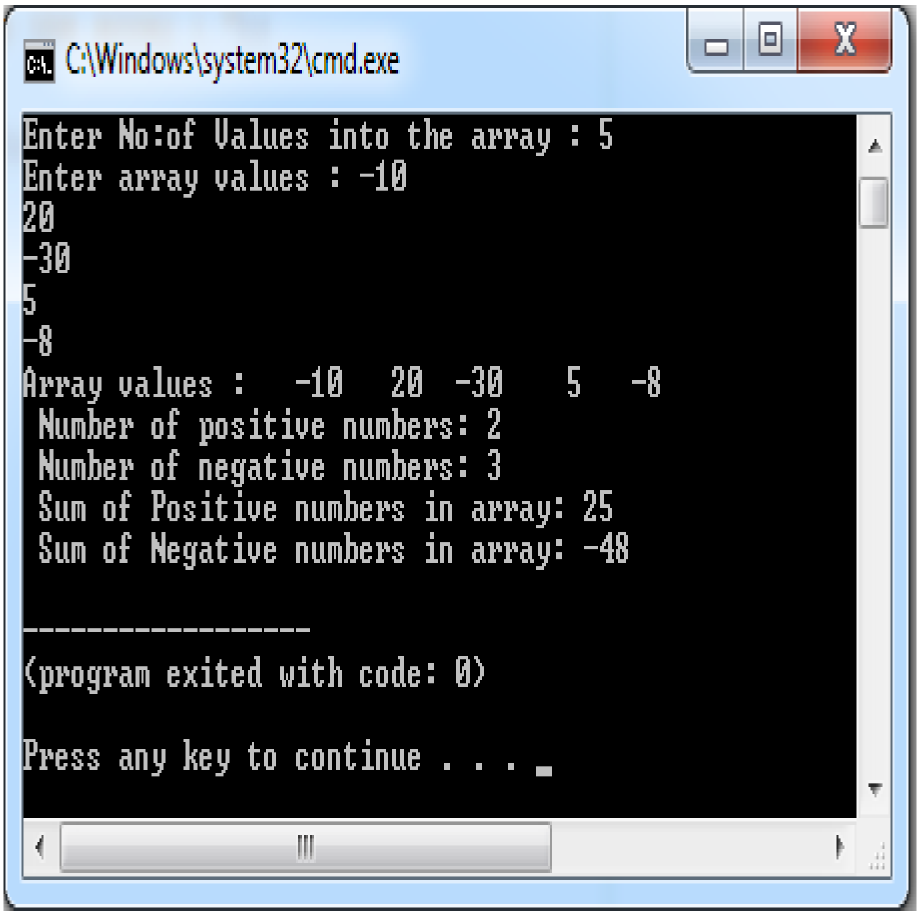
Flowchart:


start
read n
i = 0
IS i < n ?
False
True
read a[i]
i = i + 1
i = 0
IS i < n ?
True
False
print a[i]
i = i + 1
i = 0
pctr = 0
nctr = 0
psum = 0
nsum = 0
IS i < n ?
False
True
i = i + 1
print pctr
print psum
print nctr
print nsum
stop
IS a[i] > 0 ?
False
True
nctr = nctr + 1
nsum = nsum + a[i]
pctr = pctr + 1
psum = psum + a[i]

Program:

```
#include <stdio.h>
int main()
{
     int i, n, arr[10], pctr, nctr, psum, nsum;
     printf("Enter No:of Values into the array : ");
     scanf("%d", &n);
     printf("Enter array values : ");
     for (i = 0; i < n; i++)
          scanf("%d", &arr[i]);

     pctr = nctr = psum = nsum = 0;
     for(i = 0; i < n; i++)
     {
          if(arr[i] > 0)
          {
               pctr++;
               psum = psum + arr[i];
          }
          else
          {
               nctr++;
               nsum = nsum + arr[i];
          }
     }

     printf("Array values : ");
     for (i = 0; i < n; i++)
          printf("%5d", arr[i]);

     printf("\n Number of positive numbers: %d", pctr);
     printf("\n Number of negative numbers: %d", nctr);
     printf("\n Sum of Positive numbers in array: %d", psum);
     printf("\n Sum of Negative numbers in array: %d", nsum);
     return 0;
}
```

Output:



```
Enter No:of Values into the array : 5
Enter array values : -10
20
-30
5
-8
Array values :    -10   20  -30    5   -8
 Number of positive numbers: 2
 Number of negative numbers: 3
 Sum of Positive numbers in array: 25
 Sum of Negative numbers in array: -48

------------------
(program exited with code: 0)

Press any key to continue . . .
```

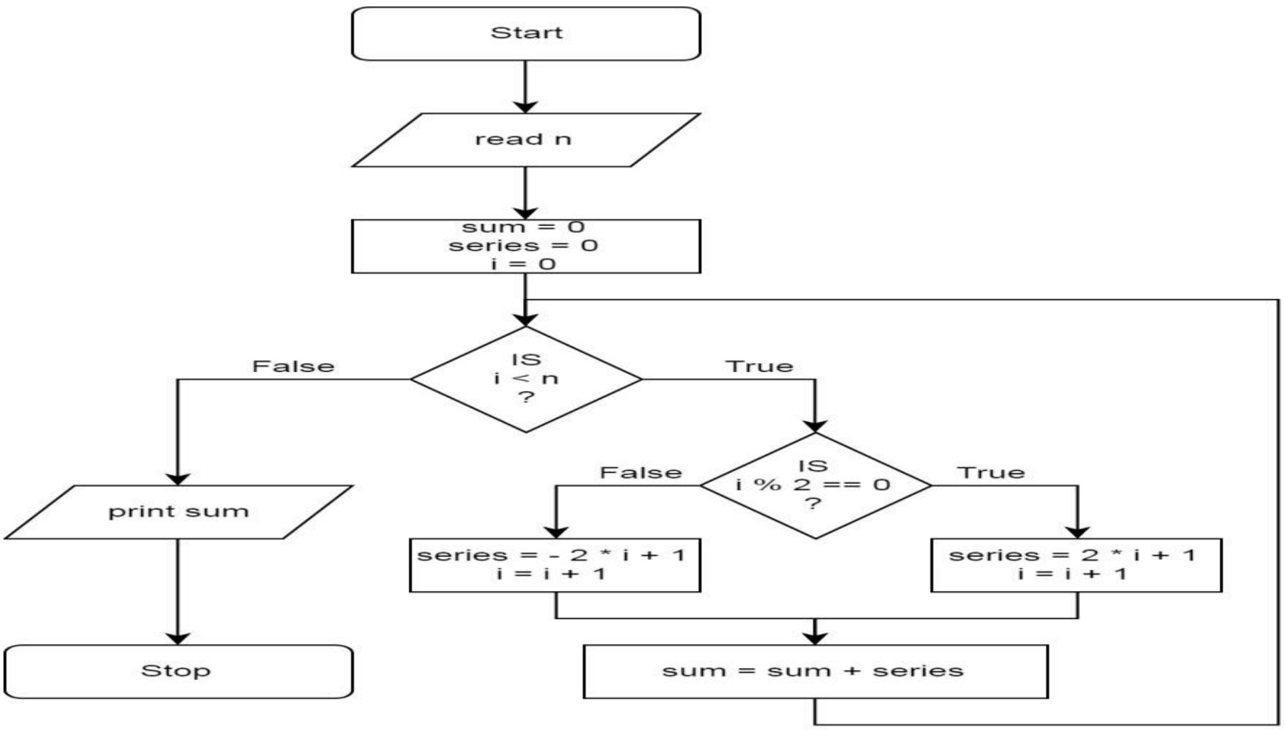
Result: The above program successfully executed.

Aim: Implement the C program which computes the sum of the first n terms of the series Sum = 1 - 3 + 5 -7 + 9

Algorithm:

```
start
print “enter the number”
input n
sum ← 0
series ← 0
i ← 0
while i < n
begin
     if i % 2 = 0 then
          series = 2 * i + 1
     else
          series = - 2 * i + 1
     sum ← sum + series
     i ← i + 1
end
print sum
stop
```

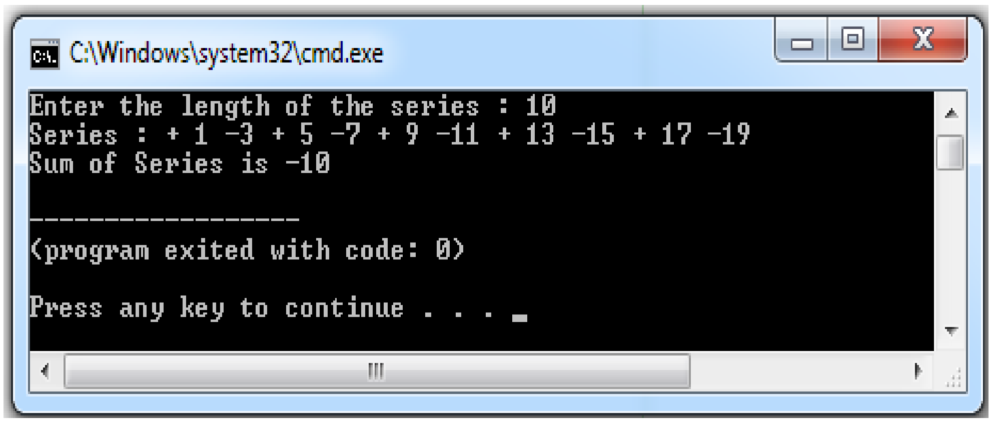
Flowchart:



Program:

```
#include<stdio.h>

int main()
{
     int i ,n, sum, series;
     printf("Enter the length of the series : ");
     scanf("%d", &n);
     sum = series = 0;
     printf("Series : ");
     for (i = 0; i < n; i++)
     {
          if ( i % 2 == 0)
          {
               series = 2 * i + 1;
               printf("+ %d ", series);
          }
          else
          {
               series = - (2 * i + 1);
               printf("%d ", series);
          }
          sum = sum + series;
     }
     printf("\nSum of Series is %d", sum);
}
```

Output:



```
Enter the length of the series : 10
Series : + 1 -3 + 5 -7 + 9 -11 + 13 -15 + 17 -19
Sum of Series is -10

------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

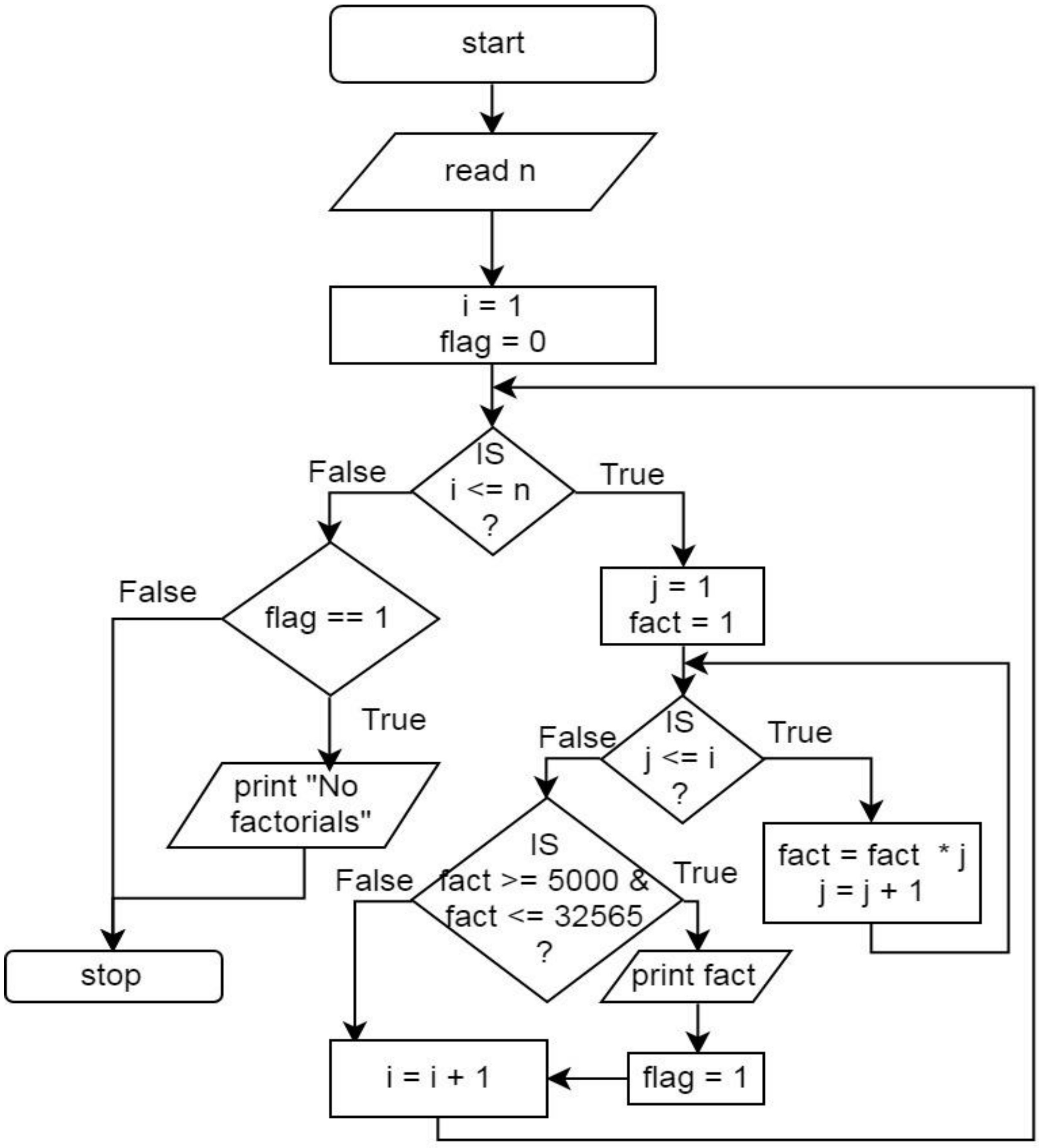
Week - 5:

Aim: Design a C program which determines the numbers whose factorial values are between 5000 and 32565.

Algorithm:

```
start
print "enter the number"
input n
flag ← 0
i ← 1
while i <= n
begin
     j ← 1
     fact ← 1
     while j <= i
     begin
          fact ← fact * j
          j ← j + 1
     end
     i ← i + 1
     if fact >= 5000 AND fact <= 32565 then
          print fact
          flag ← 1
end
if flag = 0 then
     print "No factorial values between those"
stop
```

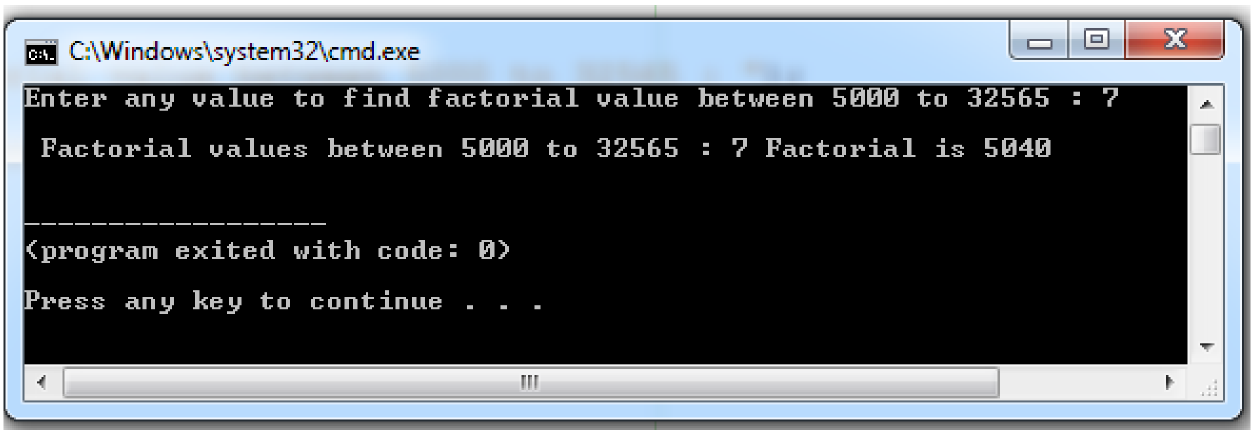
Flowchart:


start
read n
i = 1
flag = 0
IS
i <= n
?
False
True
flag == 1
False
True
print "No factorials"
stop
j = 1
fact = 1
IS
j <= i
?
False
True
fact = fact * j
j = j + 1
IS
fact >= 5000 &
fact <= 32565
?
False
True
print fact
flag = 1
i = i + 1

Program:

```
#include<stdio.h>
int main()
{
	int i, j, n, flag = 0;
	unsigned long long fact;
	printf("Enter any value to find factorial value between 5000 
to 32565 : ");
	scanf("%d", &n);
	printf("\n Factorial values between 5000 to 32565 : ");
	for(i = 1; i <= n; i++)
	{
		fact = 1;
		for(j = 1; j <= i; j++)
			fact = fact * j;
		if ( fact >= 5000 && fact <= 32565)
		{
			printf("%d Factorial is %d\n", i, fact);
			flag = 1;
		}
	}
	if ( flag == 0)
		printf("\n No Factorial Values");
}
```

Output:



```
Enter any value to find factorial value between 5000 to 32565 : 7

 Factorial values between 5000 to 32565 : 7 Factorial is 5040


------------------
(program exited with code: 0)
Press any key to continue . . .
```

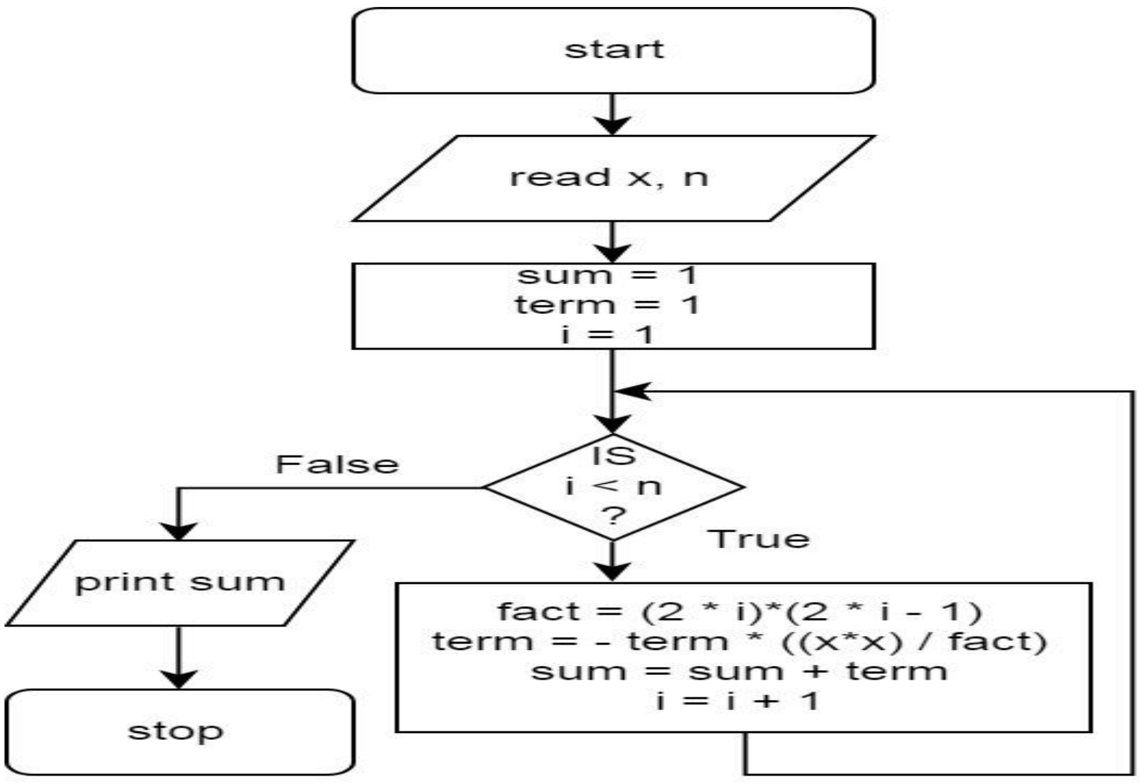
Result: The above program successfully executed.

Aim: Design an algorithm and implement using a C program which finds the sum of the infinite series $1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-......$

Algorithm:

```
start
read x, n
sum ← 1
term ← 1
i ← 1
while i < n
begin
     fact ← ( 2 * i) * ( 2 * i - 1)
     term ← -term * ((x * x)/ fact)
     sum ← sum + term
     i ← i + 1
end
print sum
stop
```

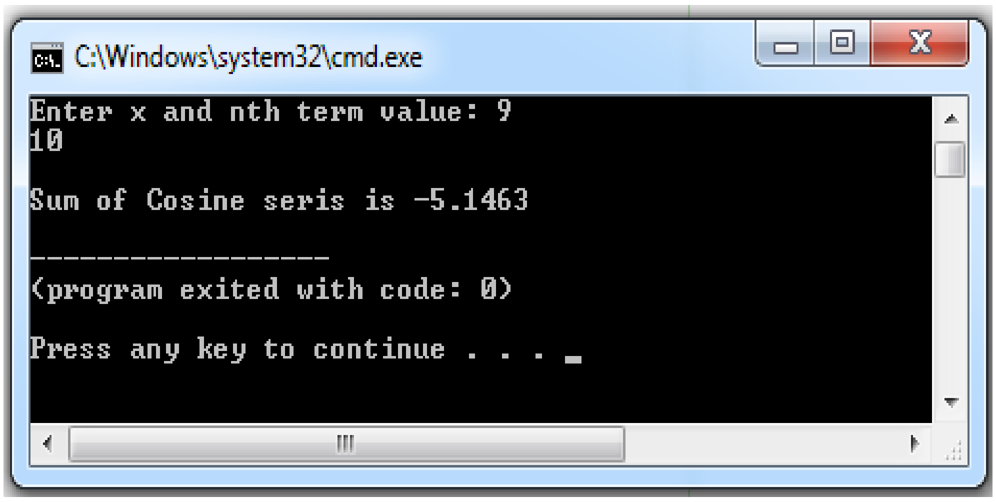
Flowchart:



Program:

```
#include <math.h>
#include <stdio.h>

int main()
{
     float sum = 1,term = 1, fact;
     int i, x, n;

     printf("Enter x and nth term value: ");
     scanf("%d%d",&x,&n);
     // Sum of n-1 terms starting from 2nd term
     for (i = 1; i < n; i++)
     {
          fact = (2*i)*(2*i-1);
          term = -term * ((x*x)/fact);
          sum = sum + term;
     }
     printf("\nSum of Cosine seris is %.4f", sum);
}
```

Output:



```
Enter x and nth term value: 9
10

Sum of Cosine seris is -5.1463

--------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

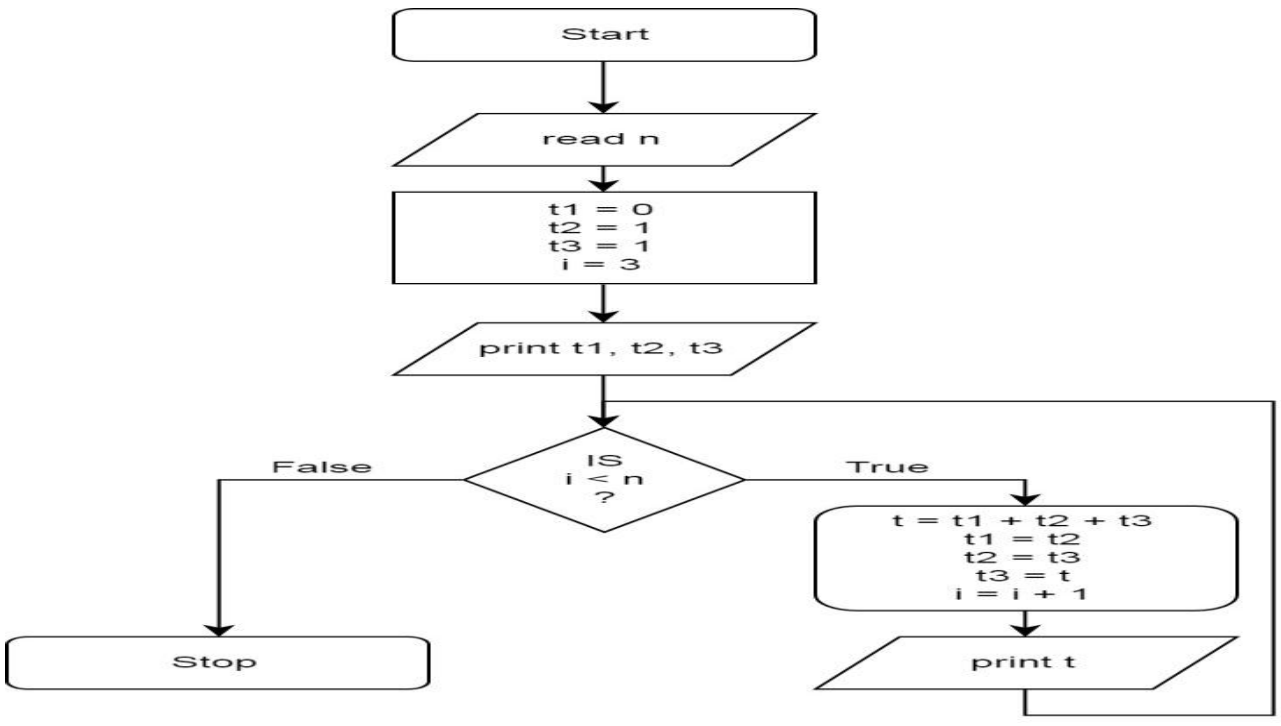
Week – 6:

Aim: Design a C program to print the sequence of numbers in which each number is the sum of the three most recent predecessors. Assume first three numbers as 0, 1, and 1.

Algorithm:

```
start
print "enter the number of terms"
input n
c ← 3
t1 ← 0
t2 ← 1
t3 ← 1
print t1, t2, t3
begin:    t ← t1 + t2 + t3
          print t
          c ← c + 1
          t1 ← t2
          t2 ← t3
          t3 ← t
          if c < n then goto begin
stop
```

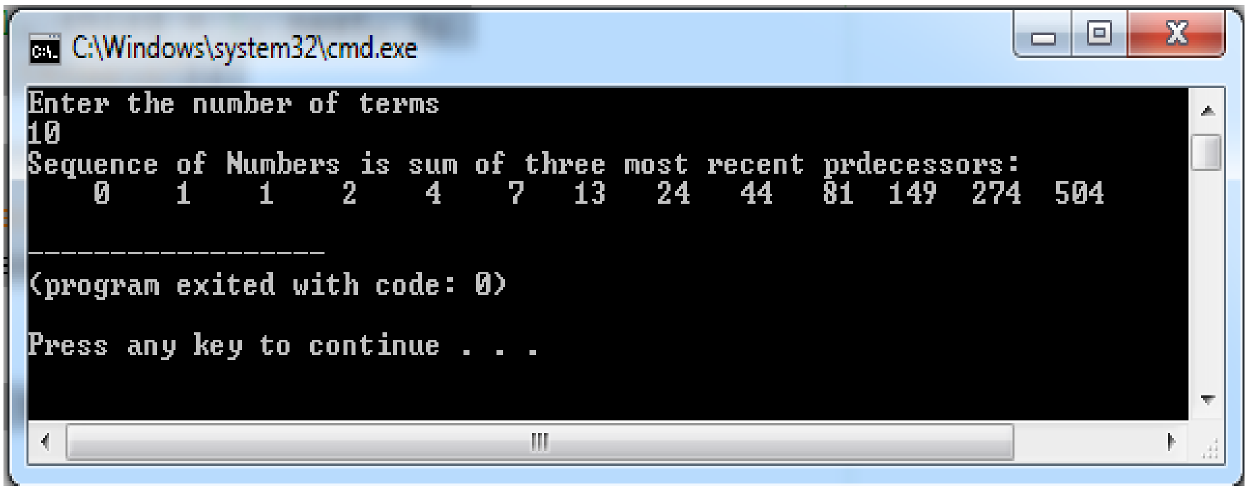
Flowchart:



Program:

```
#include <stdio.h>

int main()
{
     int n, first = 0, second = 1, third = 1, next, c;
     printf("Enter the number of terms\n");
     scanf("%d", &n);
     printf("Sequence of Numbers is ");
     printf("sum of three most recent prdecessors:\n");
     printf("%5d%5d%5d", first, second, third);
     for (c = 0; c < n; c++)
     {
          next = first + second + third;
          first = second;
          second = third;
          third = next;
          printf("%5d", next);
     }
     return 0;
}
```

Output:



```
Enter the number of terms
10
Sequence of Numbers is sum of three most recent prdecessors:
    0    1    1    2    4    7   13   24   44   81  149  274  504

------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

Week - 7:

Aim: Implement a C program which converts a hexadecimal, octal and binary number to decimal number and vice versa.

a.Hexadecimal to Decimal and Vice - Versa.

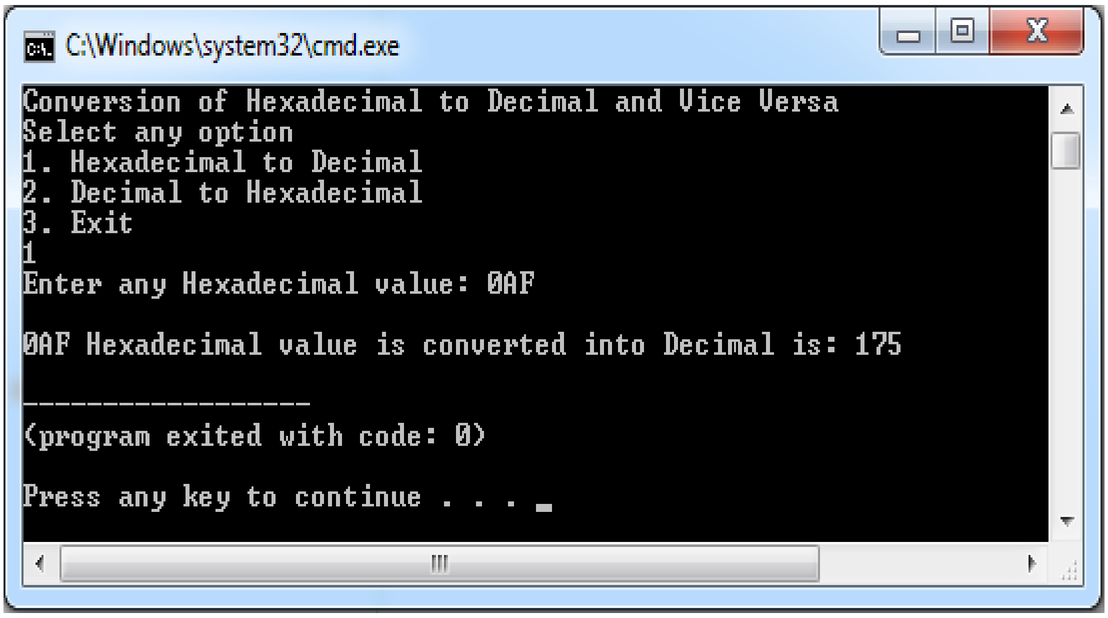
Program:

```
#include<stdio.h>
#include<string.h>

int hexaToDeci(char []);
void deciToHexa(int);

int main()
{
     int opt, decNum;
     char hexNum[20];
     printf("Conversion of Hexadecimal to Decimal & Vice Versa\n");
     printf("Select any option\n");
     printf("1. Hexadecimal to Decimal\n");
     printf("2. Decimal to Hexadecimal\n");
     printf("3. Exit\n");
     scanf("%d", &opt);
     switch(opt)
     {
          case 1:   printf("Enter any Hexadecimal value: ");
                    scanf("%s", hexNum);
                    printf("\n%s Hexadecimal value is ", hexNum);
                    printf("converted into Decimal is:");;
                    printf(" %d", hexaToDeci(hexNum)); break;
          case 2:   printf("Enter any Decimal value: ");
                    scanf("%d", &decNum);
                    printf("\n%d Decimal value is ", decNum);
                    printf("converted into HexaDecimal is: ");
                    deciToHexa(decNum); break;
          case 3:   printf("Wrong Option"); break;
     }
}
```

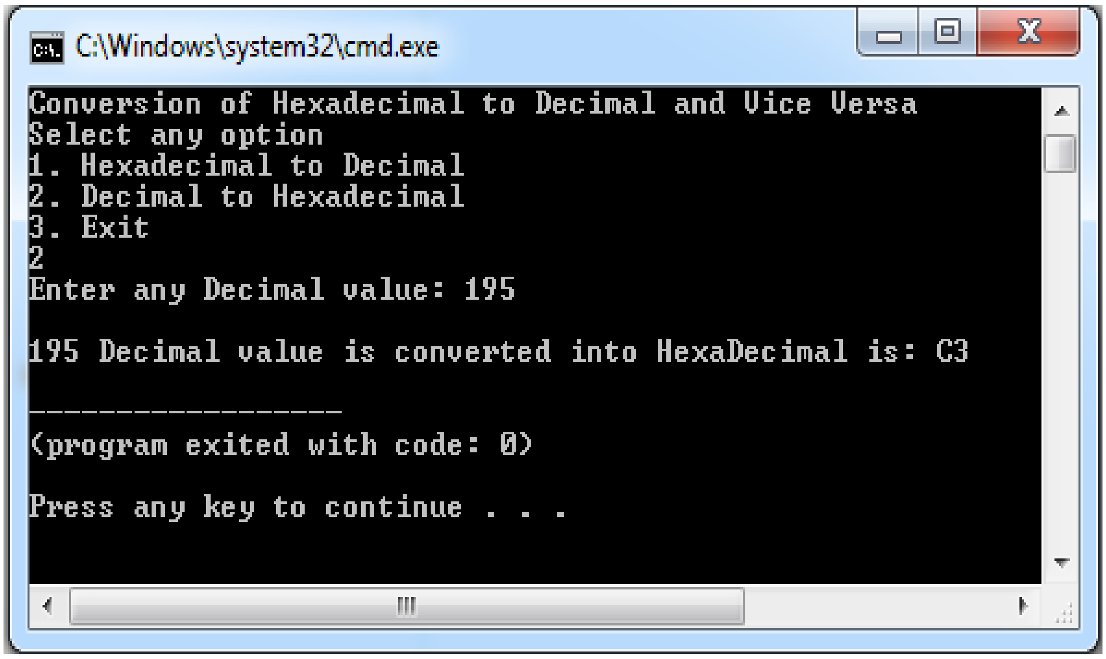
```
// Function to convert hexadecimal to decimal
int hexaToDeci(char hexVal[])
{
     int len = strlen(hexVal);

     // Initializing base value to 1, i.e 16^0
     int base = 1;
     int dec_val = 0;

     // Extracting characters as digits from last character
     for (int i=len-1; i>=0; i--)
     {
          // if character lies in '0'-'9', converting
          // it to integral 0-9 by subtracting 48 from
          // ASCII value.
          if (hexVal[i]>='0' && hexVal[i]<='9')
          {
               dec_val += (hexVal[i] - 48)*base;

               // incrementing base by power
               base = base * 16;
          }

          // if character lies in 'A'-'F' , converting
          // it to integral 10 - 15 by subtracting 55
          // from ASCII value
          else if (hexVal[i]>='A' && hexVal[i]<='F')
          {
               dec_val += (hexVal[i] - 55)*base;

               // incrementing base by power
               base = base*16;
          }
     }
     return dec_val;
}


// function to convert decimal to hexadecimal
```

```
void deciToHexa(int n)
{
     // char array to store hexadecimal number
    char hexaDeciNum[100];

    // counter for hexadecimal number array
    int i = 0;
    while(n!=0)
    {
        // temporary variable to store remainder
        int temp  = 0;

        // storing remainder in temp variable.
        temp = n % 16;

        // check if temp < 10
        if(temp < 10)
        {
            hexaDeciNum[i] = temp + 48;
            i++;
        }
        else
        {
            hexaDeciNum[i] = temp + 55;
            i++;
        }
        n = n/16;
    }

    // printing hexadecimal number array in reverse order
    for(int j=i-1; j>=0; j--)
        printf("%c", hexaDeciNum[j]);
}
```

Output:



```
Conversion of Hexadecimal to Decimal and Vice Versa
Select any option
1. Hexadecimal to Decimal
2. Decimal to Hexadecimal
3. Exit
1
Enter any Hexadecimal value: 0AF

0AF Hexadecimal value is converted into Decimal is: 175

--------------------
(program exited with code: 0)

Press any key to continue . . .
```



```
Conversion of Hexadecimal to Decimal and Vice Versa
Select any option
1. Hexadecimal to Decimal
2. Decimal to Hexadecimal
3. Exit
2
Enter any Decimal value: 195

195 Decimal value is converted into HexaDecimal is: C3

--------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

b.Octal to Decimal and Vice - Versa.

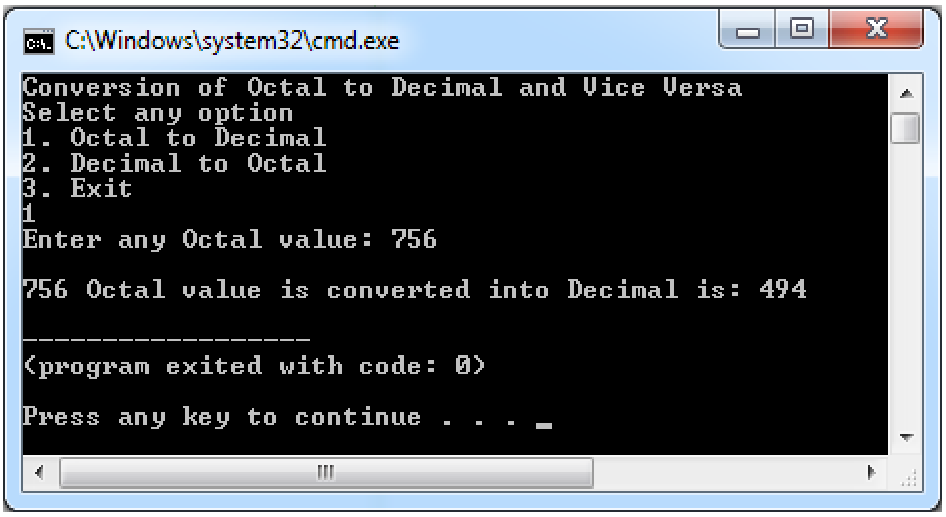
Program:

```
#include<stdio.h>
#include<string.h>

void deciToOctal(int);
int octalToDecimal(int);

int main()
{
     int opt, decNum, octNum;
     printf("Conversion of Octal to Decimal and Vice Versa\n");
     printf("Select any option\n");
     printf("1. Octal to Decimal\n");
     printf("2. Decimal to Octal\n");
     printf("3. Exit\n");
     scanf("%d", &opt);
     switch(opt)
     {
          case 1:   printf("Enter any Octal value: ");
                    scanf("%d", &octNum);
                    printf("\n%d Octal value is ", octNum);
                    printf("converted into Decimal is:");;
                    printf(" %d", octalToDecimal(octNum));
                    break;
          case 2:   printf("Enter any Decimal value: ");
                    scanf("%d", &decNum);
                    printf("\n%d Decimal value is ", decNum);
                    printf("converted into Octal is: ");
                    deciToOctal(decNum);
                    break;
          case 3:   printf("Wrong Option");
                    break;
     }

}
```

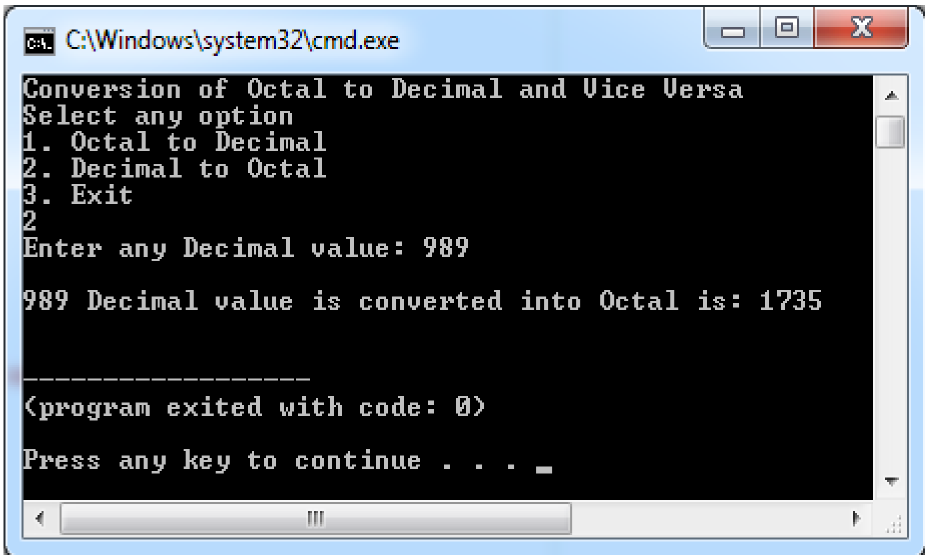
```
// Function to convert octal to decimal

int octalToDecimal(int n)
{
     int num = n;
     int dec_value = 0;

     // Initializing base value to 1, i.e 8^0
     int base = 1;
     int temp = num;
     while (temp)
     {
          // Extracting last digit
          int last_digit = temp % 10;
          temp = temp / 10;

          // Multiplying last digit with appropriate
          // base value and adding it to dec_value
          dec_value += last_digit * base;
          base = base * 8;
     }
     return dec_value;
}


// function to convert decimal to octal
void deciToOctal(int n)
{

     // array to store octal number
     int octalNum[100];

     // counter for octal number array
     int i = 0;
     while (n != 0)
     {
          // storing remainder in octal array
          octalNum[i] = n % 8;
          n = n / 8;
```

```
                i++;
        }

        // printing octal number array in reverse order
        for (int j = i - 1; j >= 0; j--)
                printf("%d", octalNum[j]);

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

Output:



```
C:\Windows\system32\cmd.exe
Conversion of Octal to Decimal and Vice Versa
Select any option
1. Octal to Decimal
2. Decimal to Octal
3. Exit
1
Enter any Octal value: 756

756 Octal value is converted into Decimal is: 494

------------------
(program exited with code: 0)

Press any key to continue . . .
```



```
C:\Windows\system32\cmd.exe
Conversion of Octal to Decimal and Vice Versa
Select any option
1. Octal to Decimal
2. Decimal to Octal
3. Exit
2
Enter any Decimal value: 989

989 Decimal value is converted into Octal is: 1735


------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

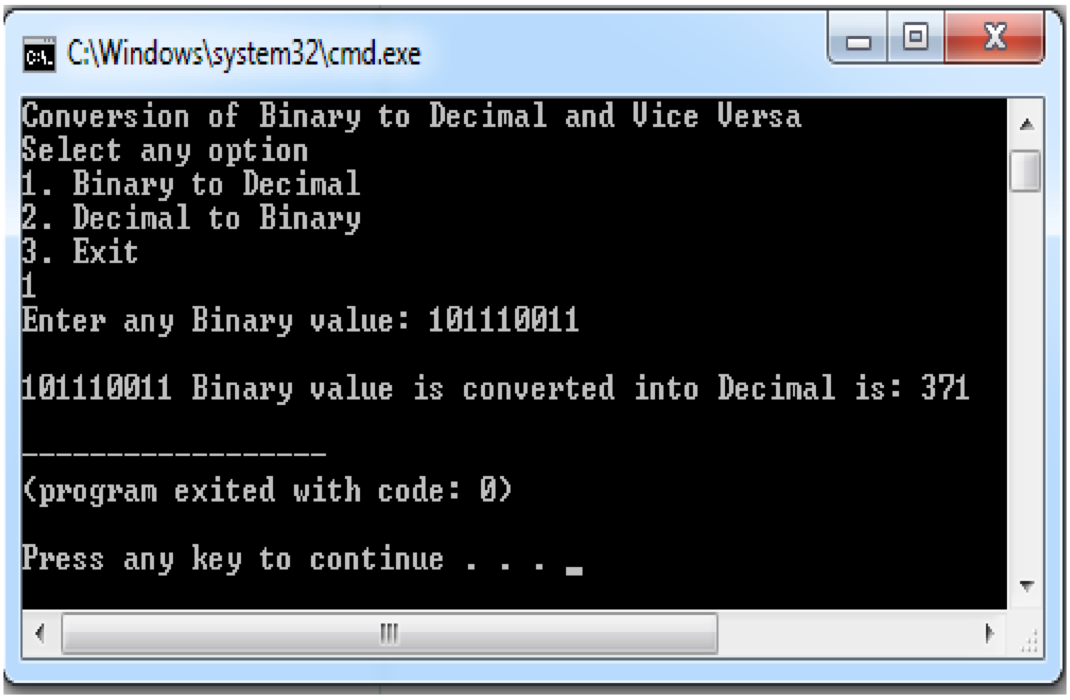
c.Binary to Decimal and Vice - Versa.

Program:

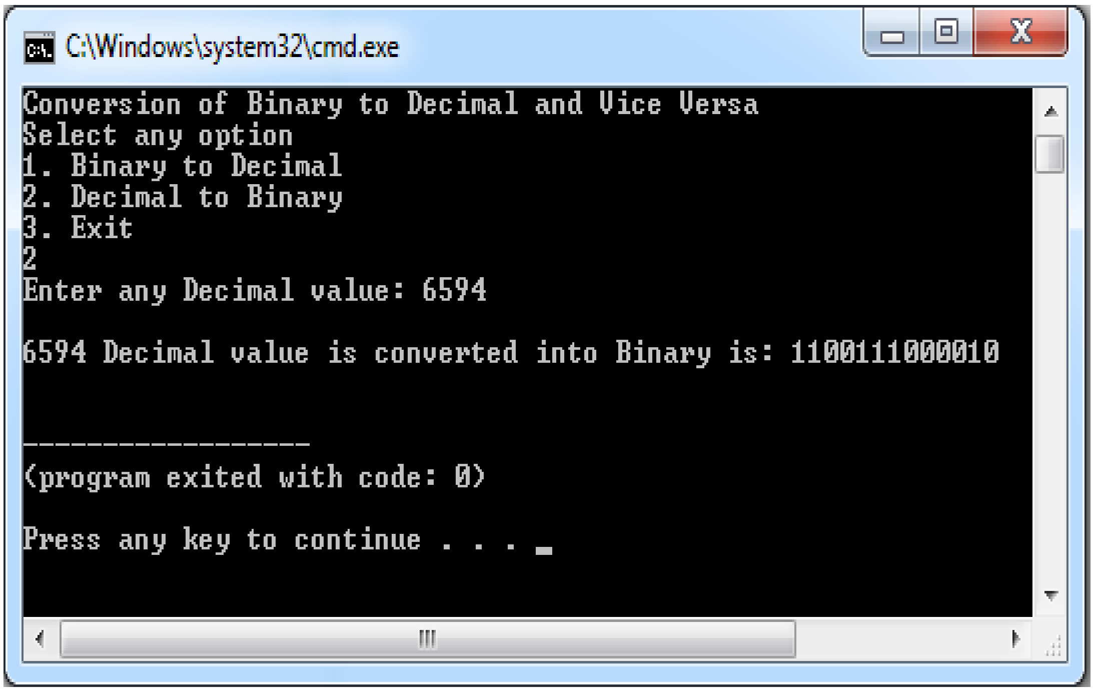
```
#include<stdio.h>
#include<string.h>

int binToDeci(char []);
void deciToBin(int);

int main()
{
     int opt, decNum;
     char binNum[20];
     printf("Conversion of Binary to Decimal and Vice Versa\n");
     printf("Select any option\n");
     printf("1. Binary to Decimal\n");
     printf("2. Decimal to Binary\n");
     printf("3. Exit\n");
     scanf("%d", &opt);
     switch(opt)
     {
          case 1:   printf("Enter any Binary value: ");
                    scanf("%s", binNum);
                    printf("\n%s Binary value is ", binNum);
                    printf("converted into Decimal is:");;
                    printf(" %d", binToDeci(binNum));
                    break;
          case 2:   printf("Enter any Decimal value: ");
                    scanf("%d", &decNum);
                    printf("\n%d Decimal value is ", decNum);
                    printf("converted into Binary is: ");
                    deciToBin(decNum);
                    break;
          case 3:   printf("Wrong Option");
                    break;
     }

}
```

```
// Function to convert binary to decimal
int binToDeci(char binValue[])
{
     int dec_value = 0;

     // Initializing base value to 1, i.e 2^0
     int base = 1;
     int len = strlen(binValue);
     for (int i = len - 1; i >= 0; i--)
     {
          if (binValue[i] == '1')
               dec_value += base;
          base = base * 2;
     }
     return dec_value;
}

// function to convert decimal to binary
void deciToBin(int n)
{
     // array to store binary number
     int binaryNum[32];

     // counter for binary array
     int i = 0;
     while (n > 0)
     {
          // storing remainder in binary array
          binaryNum[i] = n % 2;
          n = n / 2;
          i++;
     }

     // printing binary array in reverse order
     for (int j = i - 1; j >= 0; j--)
          printf("%d", binaryNum[j]);

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

Output:



```
Conversion of Binary to Decimal and Vice Versa
Select any option
1. Binary to Decimal
2. Decimal to Binary
3. Exit
1
Enter any Binary value: 101110011

101110011 Binary value is converted into Decimal is: 371

------------------
(program exited with code: 0)

Press any key to continue . . . _
```



```
Conversion of Binary to Decimal and Vice Versa
Select any option
1. Binary to Decimal
2. Decimal to Binary
3. Exit
2
Enter any Decimal value: 6594

6594 Decimal value is converted into Binary is: 1100111000010


------------------
(program exited with code: 0)

Press any key to continue . . . _
```

Result: The above program successfully executed.

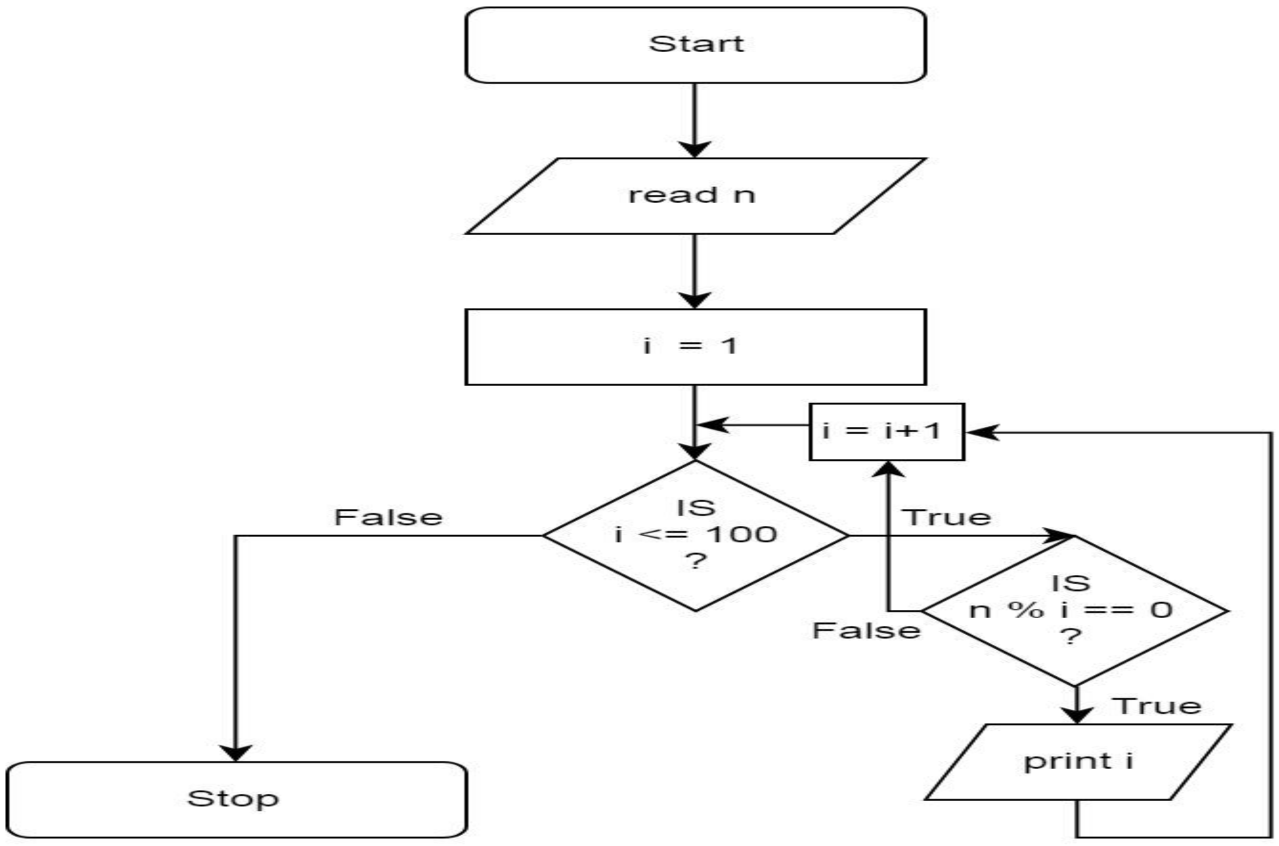
Week – 8:

Aim: Develop an algorithm which computes the all the factors between 1 to 100 for a given number and implement it using C.

Algorithm:

```
start
print "enter the number"
input n
i ← 1
while i <= 100
begin
     if n % i = 0 then
          print i
     i ← i + 1
end
stop
```

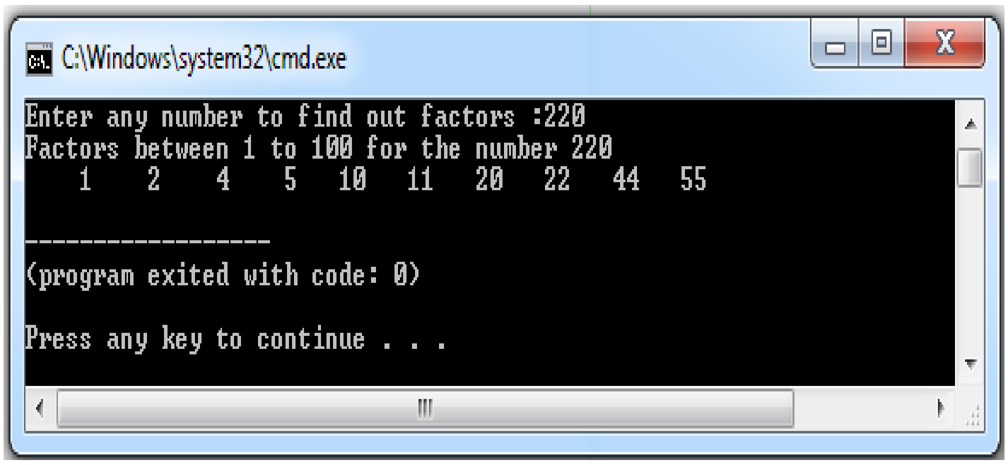
Flowchart:



Program:

```
#include<stdio.h>

int main()
{
      int n,i;
      printf("Enter any number to find out factors :");
      scanf("%d", &n);
      printf("Factors between 1 to 100 for the number %d\n", n);
      for(i = 1; i <= 100; i++)
      {
           if (n % i == 0)
           {
                printf("%5d", i);
           }
      }
}
```

Output:



```
Enter any number to find out factors :220
Factors between 1 to 100 for the number 220
    1    2    4    5   10   11   20   22   44   55

------------------
(program exited with code: 0)

Press any key to continue . . .
```

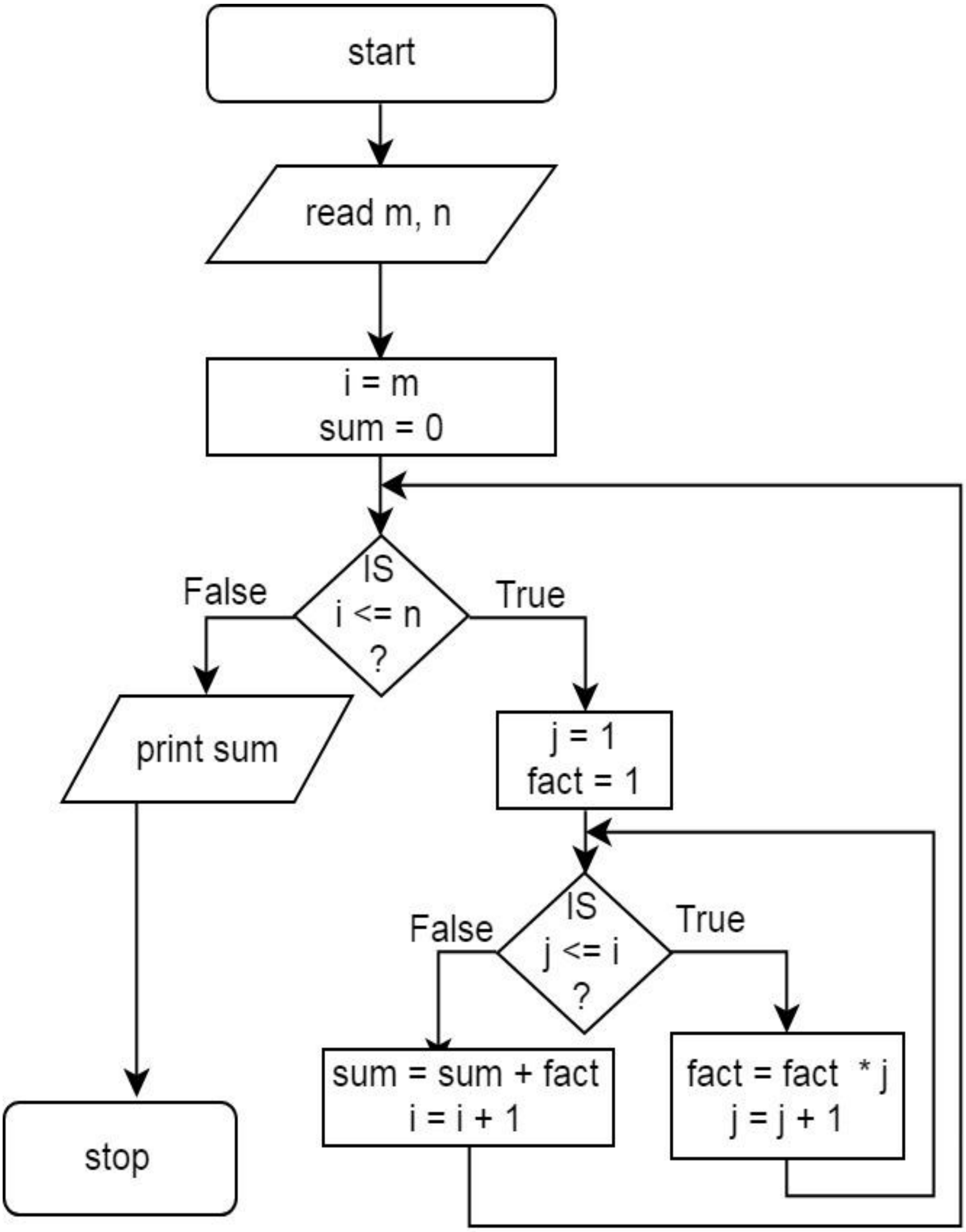
Result: The above program successfully executed.

Aim: Construct an algorithm which computes the sum of the factorials of numbers between m and n.

Algorithm:

```
start
print “enter any two number”
input m, n
i ← m
sum ← 0
while i <= n
begin
     j ← 1
     fact ← 1
     while j <= i
     begin
          fact ← fact * j
          j ← j + 1
     end
     sum ← sum + fact
     i ← i + 1
end
print sum
stop
```

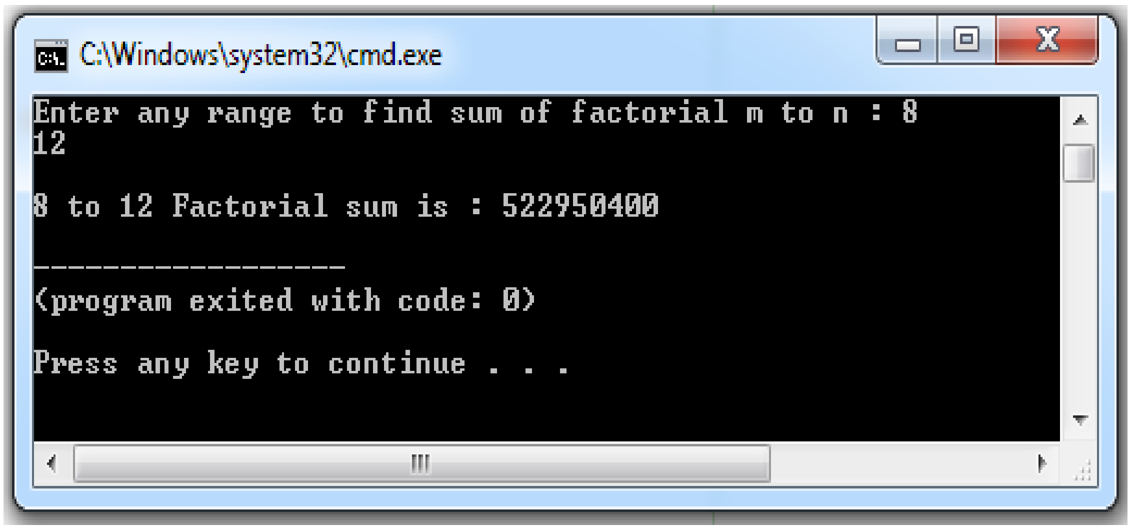
Flowchart:


start
read m, n
i = m
sum = 0
IS
i <= n
?
False
True
print sum
j = 1
fact = 1
IS
j <= i
?
False
True
sum = sum + fact
i = i + 1
fact = fact * j
j = j + 1
stop

Program:

```
#include<stdio.h>

int main()
{
      int i, j, m, n;
      unsigned long long fact, sum = 0;

      printf("Enter any range to find sum of factorial m to n : ");
      scanf("%d%d", &m, &n);

      for(i = m; i <= n; i++)
      {
            fact = 1;
            for(j = 1; j <= i; j++)
                  fact = fact * j;
            sum = sum + fact;
      }
      printf("\n%d to %d Factorial sum is : %llu", m, n, sum);
}
```

Output:



```
Enter any range to find sum of factorial m to n : 8
12

8 to 12 Factorial sum is : 522950400

--------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

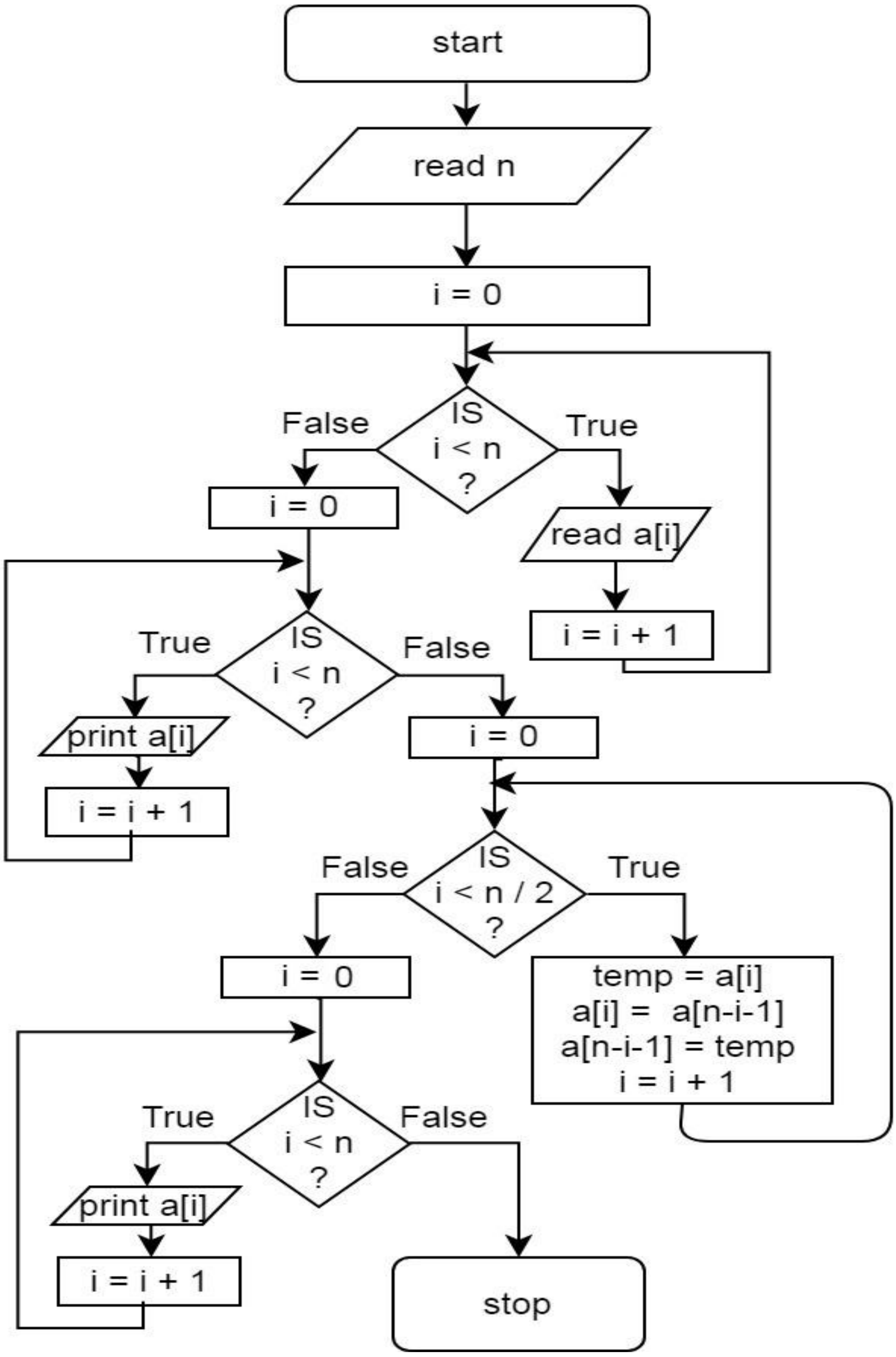
Week - 9:

Aim: Design a C program which reverses the elements of the array.

Algorithm:

```
start
print "enter the no. Of elements in the array"
input n
i ← 0
l1:  print "enter array element"
     input list[i]
     i ← i + 1
     if i<n then
          goto l1
print "array elements"
i ← 0
l2:  print list[i]
     i ← i + 1
     if i<n then
          goto l2
i ← 0
l3:  temp ← list[i]
     list[i] ← list[n-i-1]
     list[n-i-1] ← temp
     i ← i + 1
     if i<n/2 then
          goto l3
print "reversal array elements"
i ← 0
l4:  print list[i]
     i ← i + 1
     if i<n then
     goto l4
stop
```

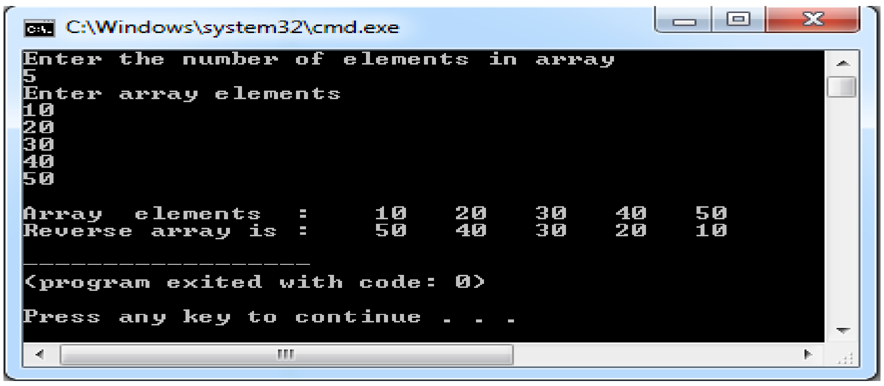
Flowchart:


start
read n
i = 0
IS i < n ?
False
True
read a[i]
i = i + 1
i = 0
IS i < n ?
True
False
print a[i]
i = i + 1
i = 0
IS i < n / 2 ?
False
True
temp = a[i]
a[i] = a[n-i-1]
a[n-i-1] = temp
i = i + 1
i = 0
IS i < n ?
True
False
print a[i]
i = i + 1
stop

Program:

```
#include <stdio.h>

int main()
{
      int n, t, c, a[100];
      printf("Enter the number of elements in array\n");
      scanf("%d", &n);
      printf("Enter array elements\n");
      for (c = 0; c < n ; c++)
            scanf("%d", &a[c]);
      printf("\nArray  elements  : ");
      for (c = 0; c < n ; c++)
            printf("%5d", a[c]);
      for(c=0;c<n/2;c++)
      {
            t = a[c];
            a[c]=a[n-c-1];
            a[n-c-1]=t;
      }
      printf("\nReverse array is : ");
      for (c = 0; c < n; c++)
            printf("%5d", a[c]);
}
```

Output:



```
Enter the number of elements in array
5
Enter array elements
10
20
30
40
50

Array  elements  :    10   20   30   40   50
Reverse array is :    50   40   30   20   10
------------------
(program exited with code: 0)
Press any key to continue . . .
```

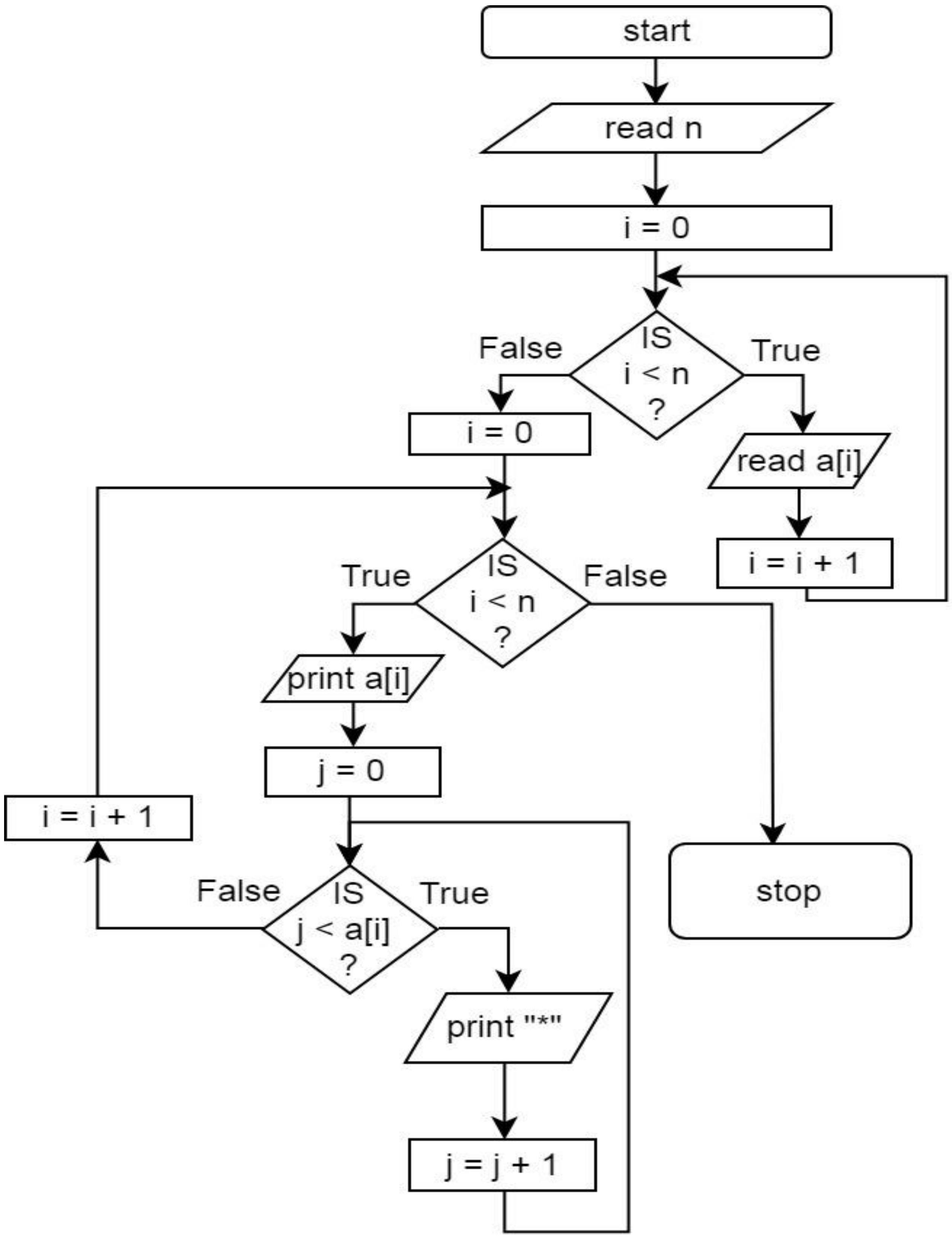
Result: The above program successfully executed.

Aim: Given a list of n numbers, Design an algorithm which prints the number of stars equivalent to the value of the number. The starts for each number should be printed horizontally.

Algorithm:

```
start
print “enter the no. Of elements in the array”
input n
i ← 0
l1:  print “enter array element”
     input list[i]
     i ← i + 1
     if i<n then
          goto l1
i ← 0
l2:  print list[i] + “        “
     j ← 0
     l3:  print “*”
          j ← j + 1
          if j < list[i] then
               goto l3
     i ← i + 1
     if i < n - 1 then
          goto l2
stop
```

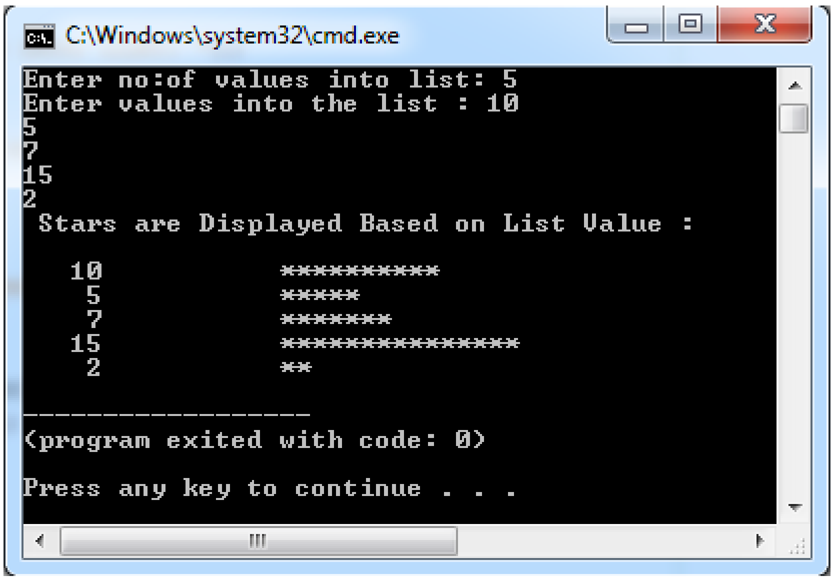
Flowchart:


start
read n
i = 0
IS i < n ?
False
True
i = 0
read a[i]
i = i + 1
IS i < n ?
True
False
print a[i]
j = 0
i = i + 1
IS j < a[i] ?
False
True
stop
print "*"
j = j + 1

Program:

```
#include <stdio.h>

int main()
{
	int i, j, n, arr[10];
	printf("Enter no:of values into list: ");
	scanf("%d", &n);
	printf("Enter values into the list : ");
	for (i = 0; i < n; i++)
		scanf("%d", &arr[i]);
	printf(" Stars are Displayed Based on List Value : \n");
	for(i = 0; i < n; i++)
	{
		printf("\n%5d\t\t", arr[i]);
		for (j = 0; j < arr[i]; j++)
			printf("*");
	}
}
```

Output:



```
Enter no:of values into list: 5
Enter values into the list : 10
5
7
15
2
 Stars are Displayed Based on List Value :

   10           **********
    5           *****
    7           *******
   15           ***************
    2           **

------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

Week - 10:

Aim: Implement the sorting algorithms

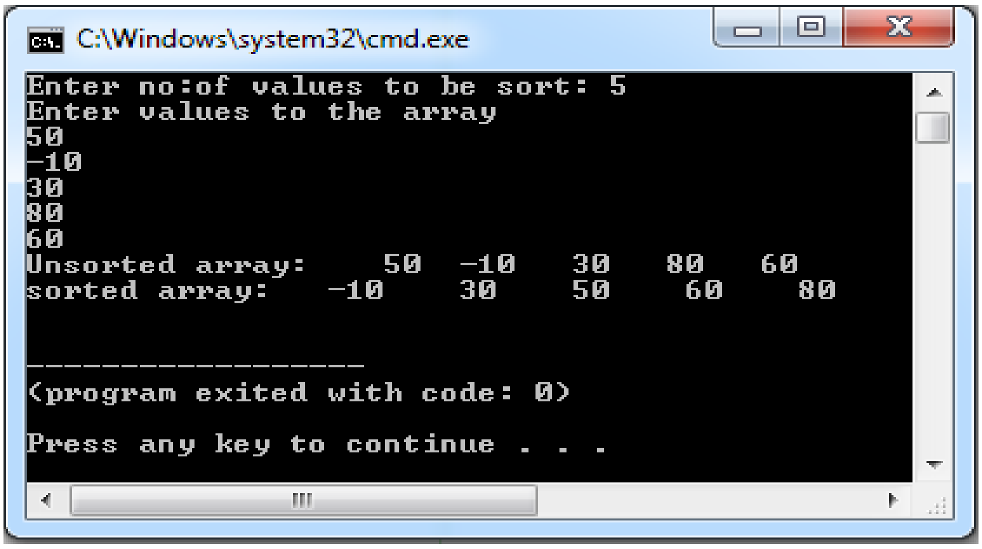
i. Insertion sort

Program:

```
#include <stdio.h>

Void insertionSort(int [], int);
Void printArray(int[], int);

int main()
{
     int arr[20], i, n;

     printf("Enter no:of values to be sort: ");
     scanf("%d", &n);

     printf("Enter values to the array\n");
     for(i = 0; i < n; i++)
          scanf("%d", &arr[i]);

     printf("Unsorted array: ");
     for(i = 0; i < n; i++)
          printf("%5d", arr[i]);

     insertionSort(arr, n);
     printf("\nsorted array: ");
     printArray(arr, n);

     return 0;
}
```

```
/* Function to sort an array using insertion sort*/
void insertionSort(int arr[], int n)
{
     int i, key, j;
     for (i = 1; i < n; i++)
     {
          key = arr[i];
          j = i - 1;
          while (j >= 0 && arr[j] > key)
          {
               arr[j + 1] = arr[j];
               j = j - 1;
          }
          arr[j + 1] = key;
     }
}

void printArray(int arr[], int n)
{
     int i;
     for (i = 0; i < n; i++)
          printf("%5d ", arr[i]);
     printf("\n");
}
```

Output:



```
Enter no:of values to be sort: 5
Enter values to the array
50
-10
30
80
60
Unsorted array:    50  -10    30    80    60
sorted array:   -10    30    50    60    80


-------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

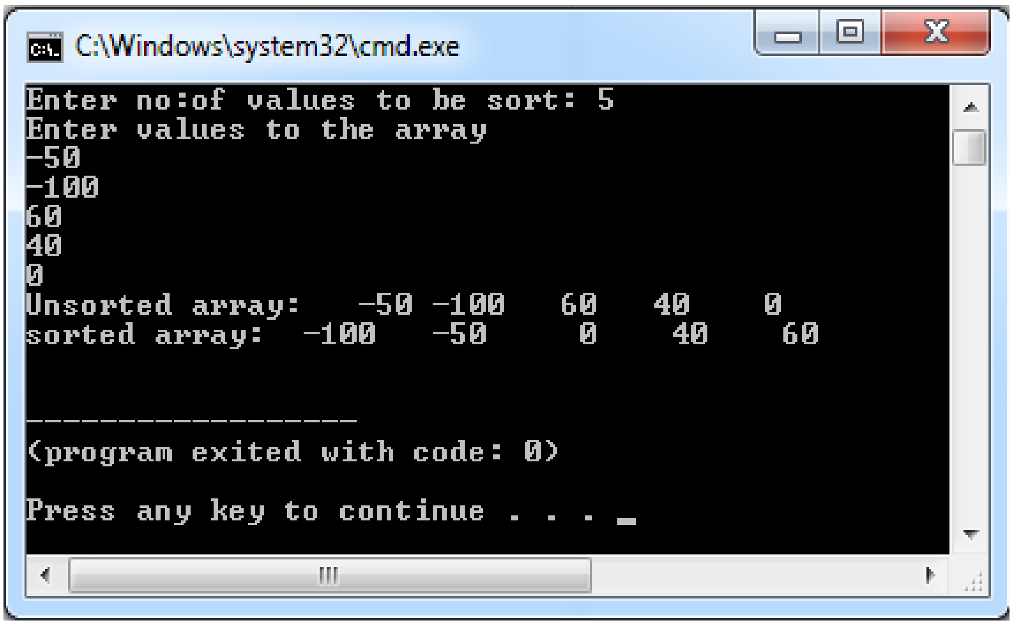
ii. Exchange sort

Program:

```
#include <stdio.h>

void exchangeSort( int [], int );
void printArray(int [], int);

int main()
{
     int arr[20], i, n;

     printf("Enter no:of values to be sort: ");
     scanf("%d", &n);

     printf("Enter values to the array\n");
     for(i = 0; i < n; i++)
          scanf("%d", &arr[i]);

     printf("Unsorted array: ");
     for(i = 0; i < n; i++)
          printf("%5d", arr[i]);

     exchangeSort(arr, n);
     printf("\nsorted array: ");
     printArray(arr, n);

     return 0;
}

void exchangeSort( int a[], int elements )
{
     int i, j, temp;
     i = 0;
     while( i < (elements - 1) )
     {
          j = i + 1;
          while( j < elements )
```

```
            {
                  if( a[i] > a[j] )
                  {
                        temp = a[i];
                        a[i] = a[j];
                        a[j] = temp;
                  }
                  j++;
            }
            i++;
      }
}

void printArray(int arr[], int n)
{
      int i;
      for (i = 0; i < n; i++)
            printf("%5d ", arr[i]);
      printf("\n");
}
```

Output:



```
Enter no:of values to be sort: 5
Enter values to the array
-50
-100
60
40
0
Unsorted array:   -50 -100   60   40    0
sorted array:  -100   -50     0    40    60


------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

Week - 11:

Aim: Implement the sorting algorithms

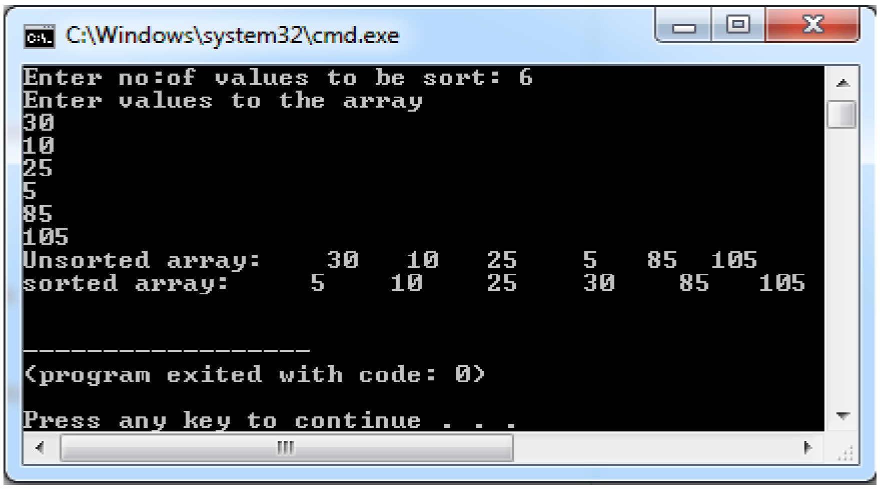
i. Selection sort

Program:

```
#include <stdio.h>
void swap(int *, int *);
void selectionSort(int [], int);
void printArray(int [], int);

void main()
{
     int arr[20], i, n;

     printf("Enter no:of values to be sort: ");
     scanf("%d", &n);

     printf("Enter values to the array\n");
     for(i = 0; i < n; i++)
          scanf("%d", &arr[i]);

     printf("Unsorted array: ");
     for(i = 0; i < n; i++)
          printf("%5d", arr[i]);

     selectionSort(arr, n);
     printf("\nsorted array: ");
     printArray(arr, n);
}

void swap(int *xp, int *yp)
{
     int temp = *xp;
     *xp = *yp;
     *yp = temp;
}
```

```
void selectionSort(int arr[], int n)
{
     int i, j, min_idx;
     // One by one move boundary of unsorted subarray
     for (i = 0; i < n-1; i++)
     {
          // Find the minimum element in unsorted array
          min_idx = i;
          for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
               min_idx = j;

          // Swap the found minimum element with the first element
          swap(&arr[min_idx], &arr[i]);
     }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
     int i;
     for (i=0; i < size; i++)
          printf("%5d ", arr[i]);
     printf("\n");
}
```

Output:



```
Enter no:of values to be sort: 6
Enter values to the array
30
10
25
5
85
105
Unsorted array:    30    10    25    5    85   105
sorted array:     5    10    25    30    85   105


------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

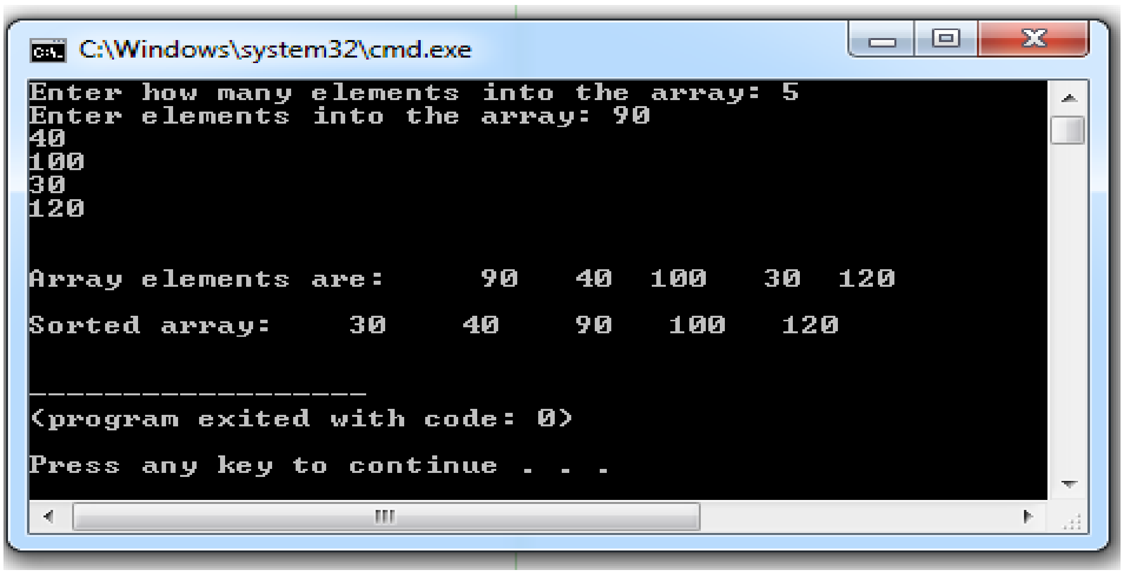
ii. Partitioning sort.

Program:

```
#include<stdio.h>
void swap(int *,int *);
int partition (int [], int, int);
void quickSort(int [], int, int);
void printArray(int [], int);

int main()
{
     int arr[20], i, n;
     printf("Enter how many elements into the array: ");
     scanf("%d", &n);
     printf("Enter elements into the array: ");
     for(i = 0; i < n; i++)
          scanf("%d", &arr[i]);
     printf("\n\nArray elements are:  ");
     for(i = 0; i < n; i++)
          printf("%5d", arr[i]);
     quickSort(arr, 0, n-1);
     printf("\n\nSorted array: ");
     printArray(arr, n);
     return 0;
}

void swap(int* a, int* b)
{
     int t = *a;
     *a = *b;
     *b = t;
}

int partition (int arr[], int low, int high)
{
     int pivot = arr[high]; // pivot
     int i = (low - 1); // Index of smaller element
     for (int j = low; j <= high- 1; j++)
     {
          if (arr[j] < pivot)
```

```
            {
                  i++; // increment index of smaller element
                  swap(&arr[i], &arr[j]);
            }
      }
      swap(&arr[i + 1], &arr[high]);
      return (i + 1);
}

void quickSort(int arr[], int low, int high)
{
      if (low < high)
      {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
      }
}

void printArray(int arr[], int size)
{
      int i;
      for (i=0; i < size; i++)
            printf("%5d ", arr[i]);
      printf("\n");
}
```

Output:



```
Enter how many elements into the array: 5
Enter elements into the array: 90
40
100
30
120


Array elements are:     90    40   100    30   120

Sorted array:     30     40     90    100    120


--------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

Week - 12:

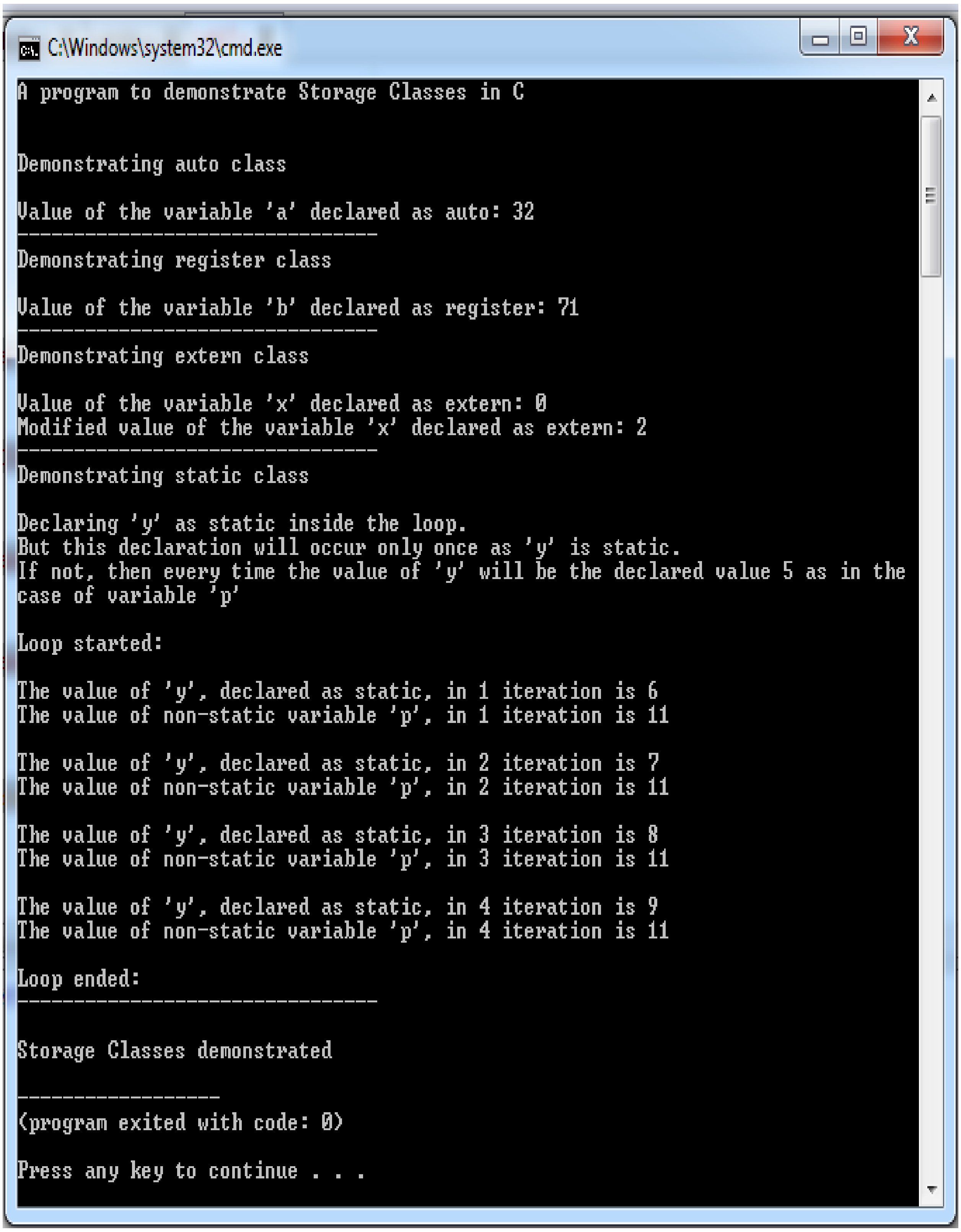
Aim: Illustrate the use of auto, static, register and external variables.

Program:

```
// A C program to demonstrate different storage classes

#include <stdio.h>

void autoStorageClass();
void registerStorageClass();
void externStorageClass();
void staticStorageClass();

// declaring the variable which is to be made extern
// an intial value can also be initialized to x
int x;

void main()
{

     printf("A program to demonstrate"
          " Storage Classes in C\n\n");

     // To demonstrate auto Storage Class
     autoStorageClass();

     // To demonstrate register Storage Class
     registerStorageClass();

     // To demonstrate extern Storage Class
     externStorageClass();

     // To demonstrate static Storage Class
     staticStorageClass();

     // exiting
     printf("\n\nStorage Classes demonstrated");
}

void autoStorageClass()
{

     printf("\nDemonstrating auto class\n\n");

     // declaring an auto variable (simply
     // writing "int a=32;" works as well)
     auto int a = 32;

     // printing the auto variable 'a'
     printf("Value of the variable 'a'"
          " declared as auto: %d\n",
          a);

     printf("--------------------------------");
}

void registerStorageClass()
{

     printf("\nDemonstrating register class\n\n");

     // declaring a register variable
     register char b = 'G';

     // printing the register variable 'b'
     printf("Value of the variable 'b'"
          " declared as register: %d\n",
          b);

     printf("--------------------------------");
}

void externStorageClass()
{
     printf("\nDemonstrating extern class\n\n");

     // telling the compiler that the variable
     // z is an extern variable and has been
     // defined elsewhere (above the main
     // function)
     extern int x;

     // printing the extern variables 'x'
     printf("Value of the variable 'x'"
          " declared as extern: %d\n",
          x);

     // value of extern variable x modified
     x = 2;

     // printing the modified values of
     // extern variables 'x'
     printf("Modified value of the variable 'x'"
          " declared as extern: %d\n",
          x);

     printf("--------------------------------");
}

void staticStorageClass()
{
     int i = 0;

     printf("\nDemonstrating static class\n\n");

     // using a static variable 'y'
     printf("Declaring 'y' as static inside the loop.\n"
          "But this declaration will occur only"
          " once as 'y' is static.\n"
          "If not, then every time the value of 'y' "
          "will be the declared value 5"
          " as in the case of variable 'p'\n");
```

```
    printf("\nLoop started:\n");

    for (i = 1; i < 5; i++)
    {
         // Declaring the static variable 'y'
         static int y = 5;

         // Declare a non-static variable 'p'
         int p = 10;

         // Incrementing the value of y and p by 1
         y++;
         p++;

         // printing value of y at each iteration
         printf("\nThe value of 'y', "
              "declared as static, in %d "
              "iteration is %d\n",
              i, y);

         // printing value of p at each iteration
         printf("The value of non-static variable 'p', "
              "in %d iteration is %d\n",
              i, p);
    }

    printf("\nLoop ended:\n");

    printf("--------------------------------");
}
```

Output:



```
A program to demonstrate Storage Classes in C


Demonstrating auto class

Value of the variable 'a' declared as auto: 32
--------------------------------
Demonstrating register class

Value of the variable 'b' declared as register: 71
--------------------------------
Demonstrating extern class

Value of the variable 'x' declared as extern: 0
Modified value of the variable 'x' declared as extern: 2
--------------------------------
Demonstrating static class

Declaring 'y' as static inside the loop.
But this declaration will occur only once as 'y' is static.
If not, then every time the value of 'y' will be the declared value 5 as in the
case of variable 'p'

Loop started:

The value of 'y', declared as static, in 1 iteration is 6
The value of non-static variable 'p', in 1 iteration is 11

The value of 'y', declared as static, in 2 iteration is 7
The value of non-static variable 'p', in 2 iteration is 11

The value of 'y', declared as static, in 3 iteration is 8
The value of non-static variable 'p', in 3 iteration is 11

The value of 'y', declared as static, in 4 iteration is 9
The value of non-static variable 'p', in 4 iteration is 11

Loop ended:
--------------------------------


Storage Classes demonstrated

------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

Week - 13:

Aim: Design algorithm and implement the operations creation, insertion, deletion, traversing on a singly linked list.

Program:

```
/*
 * C Program to Implement Singly Linked List using Dynamic Memory
Allocation
 */
#include <stdio.h>
#include <malloc.h>
#define ISEMPTY printf("\nEMPTY LIST:");
/*
 * Node Declaration
 */
struct node
{
    int value;
    struct node *next;
};

typedef struct node snode;
snode *newnode, *ptr, *prev, *temp;
snode *first = NULL, *last = NULL;

snode* create_node(int);
void insert_node_first();
void insert_node_last();
void insert_node_pos();
void sorted_ascend();
void delete_pos();
void search();
void update_val();
void display();
void rev_display(snode *);
```

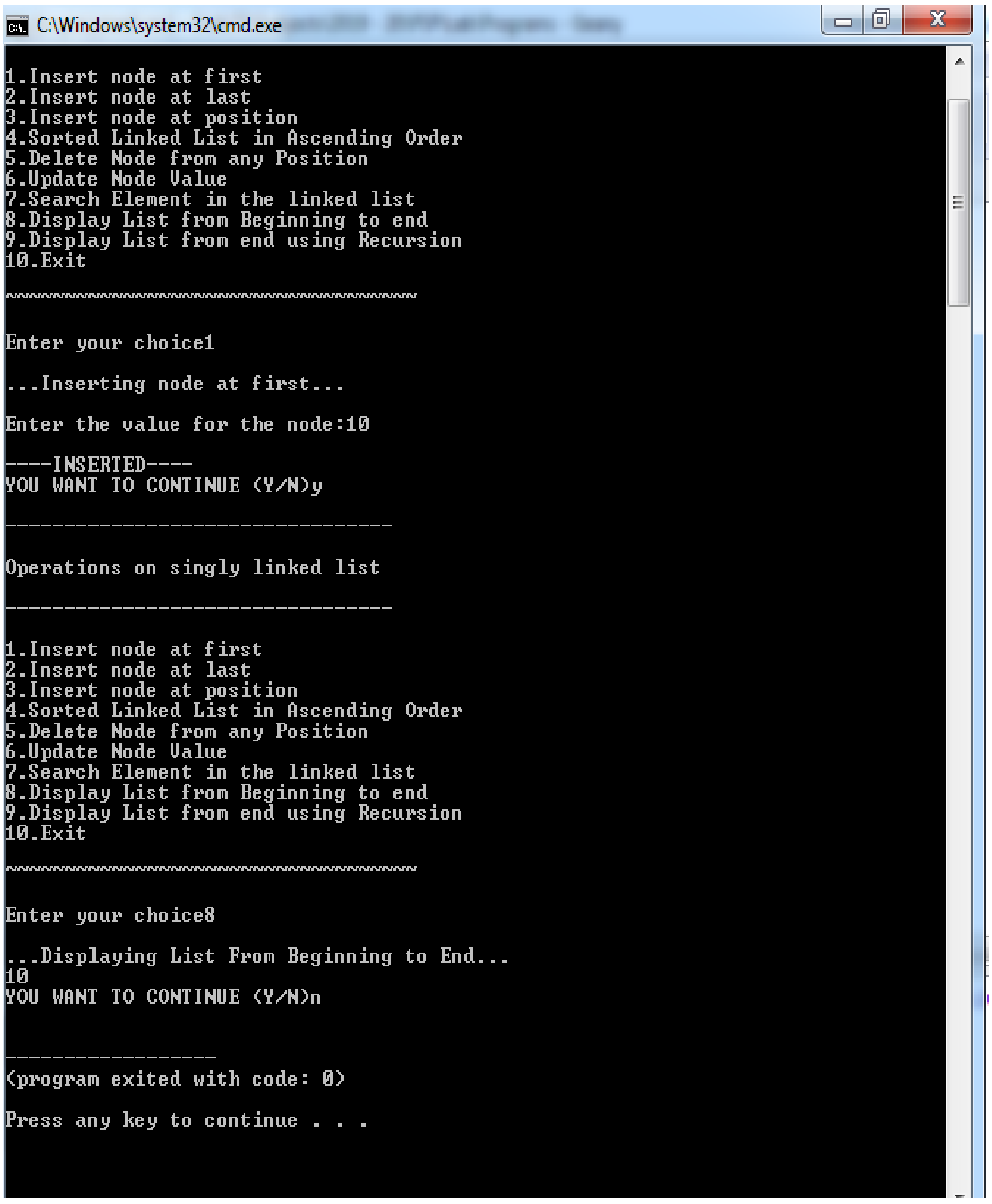
```
/*
 * Main :contains menu
 */

 int main()
 {
    int ch;
    char ans = 'Y';

    while (ans == 'Y'||ans == 'y')
    {
        printf("\n---------------------------------\n");
        printf("\nOperations on singly linked list\n");
        printf("\n---------------------------------\n");
        printf("\n1.Insert node at first");
        printf("\n2.Insert node at last");
        printf("\n3.Insert node at position");
        printf("\n4.Sorted Linked List in Ascending Order");
        printf("\n5.Delete Node from any Position");
        printf("\n6.Update Node Value");
        printf("\n7.Search Element in the linked list");
        printf("\n8.Display List from Beginning to end");
        printf("\n9.Display List from end using Recursion");
        printf("\n10.Exit\n");
        printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
        printf("\nEnter your choice");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            printf("\n...Inserting node at first...\n");
            insert_node_first();
            break;
        case 2:
            printf("\n...Inserting node at last...\n");
            insert_node_last();
```

```
            break;
        case 3:
            printf("\n...Inserting node at position...\n");
            insert_node_pos();
            break;
        case 4:
            printf("\n...Sorted    Linked    List    in    Ascending
Order...\n");
            sorted_ascend();
            break;
        case 5:
            printf("\n...Deleting Node from any Position...\n");
            delete_pos();
            break;
        case 6:
            printf("\n...Updating Node Value...\n");
            update_val();
            break;
        case 7:
            printf("\n...Searching Element in the List...\n");
            search();
            break;
        case 8:
            printf("\n...Displaying   List   From   Beginning   to
End...\n");
            display();
            break;
        case 9:
            printf("\n...Displaying    List    From    End    using
Recursion...\n");
            rev_display(first);
            break;
        case 10:
            printf("\n...Exiting...\n");
            return 0;
            break;
        default:
            printf("\n...Invalid Choice...\n");
            break;
        }
        printf("\nYOU WANT TO CONTINUE (Y/N)");
        scanf(" %c", &ans);
    }
    return 0;
 }

/*
 * Creating Node
 */
snode* create_node(int val)
{
    newnode = (snode *)malloc(sizeof(snode));
    if (newnode == NULL)
    {
        printf("\nMemory was not allocated");
        return 0;
    }
    else
    {
        newnode->value = val;
        newnode->next = NULL;
        return newnode;
    }
}

/*
 * Inserting Node at First
 */
void insert_node_first()
{
    int val;

    printf("\nEnter the value for the node:");
    scanf("%d", &val);
    newnode = create_node(val);
    if (first == last && first == NULL)
    {
        first = last = newnode;
        first->next = NULL;
        last->next = NULL;
    }
    else
    {
        temp = first;
        first = newnode;
        first->next = temp;
    }
    printf("\n----INSERTED----");
}

/*
 * Inserting Node at Last
 */
void insert_node_last()
{
    int val;

    printf("\nEnter the value for the Node:");
    scanf("%d", &val);
    newnode = create_node(val);
    if (first == last && last == NULL)
    {
        first = last = newnode;
        first->next = NULL;
        last->next = NULL;
    }
    else
    {
        last->next = newnode;
        last = newnode;
        last->next = NULL;
    }
 printf("\n----INSERTED----");
}

/*
 * Inserting Node at position
 */
```

```
void insert_node_pos()
{
    int pos, val, cnt = 0, i;

    printf("\nEnter the value for the Node:");
    scanf("%d", &val);
    newnode = create_node(val);
     printf("\nEnter the position ");
    scanf("%d", &pos);
    ptr = first;
    while (ptr != NULL)
    {
        ptr = ptr->next;
        cnt++;
    }
    if (pos == 1)
    {
        if (first == last && first == NULL)
        {
            first = last = newnode;
            first->next = NULL;
            last->next = NULL;
        }
        else
        {
            temp = first;
            first = newnode;
            first->next = temp;
        }
        printf("\nInserted");
    }
    else if (pos>1 && pos<=cnt)
    {
        ptr = first;
        for (i = 1;i < pos;i++)
        {
            prev = ptr;
            ptr = ptr->next;
        }
        prev->next = newnode;
        newnode->next = ptr;
        printf("\n----INSERTED----");
    }
    else
    {
        printf("Position is out of range");
    }
}

/*
 * Sorted Linked List
 */
void sorted_ascend()
{
    snode *nxt;
    int t;

    if (first == NULL)
    {
        ISEMPTY;
        printf(":No elements to sort\n");
    }
    else
    {
        for (ptr = first;ptr != NULL;ptr = ptr->next)
        {
            for (nxt = ptr->next;nxt != NULL;nxt = nxt->next)
            {
                if (ptr->value > nxt->value)
                {
                    t = ptr->value;
                    ptr->value = nxt->value;
                    nxt->value = t;
                }
            }
        }
        printf("\n---Sorted List---");
        for (ptr = first;ptr != NULL;ptr = ptr->next)
        {
            printf("%d\t", ptr->value);
        }
    }
}

/*
 * Delete Node from specified position in a non-empty list
 */
void delete_pos()
{
    int pos, cnt = 0, i;

    if (first == NULL)
    {
        ISEMPTY;
        printf(":No node to delete\n");
    }
    else
    {
        printf("\nEnter the position of value to be deleted:");
        scanf(" %d", &pos);
        ptr = first;
        if (pos == 1)
        {
            first = ptr->next;
            printf("\nElement deleted");
        }
        else
        {
            while (ptr != NULL)
            {
                ptr = ptr->next;
                cnt = cnt + 1;
            }
            if (pos > 0 && pos <= cnt)
            {
                ptr = first;
                for (i = 1;i < pos;i++)
                {
                    prev = ptr;
                    ptr = ptr->next;
                }
                prev->next = ptr->next;
            }
            else
            {
                printf("Position is out of range");
            }
        free(ptr);
        printf("\nElement deleted");
        }
    }
}
/*
 * Updating Node value in a non-empty list
 */
void update_val()
{
    int oldval, newval, flag = 0;

    if (first == NULL)
    {
        ISEMPTY;
        printf(":No nodes in the list to update\n");
    }
    else
    {
        printf("\nEnter the value to be updated:");
        scanf("%d", &oldval);
        printf("\nEnter the newvalue:");
        scanf("%d", &newval);
        for (ptr = first;ptr != NULL;ptr = ptr->next)
        {
            if (ptr->value == oldval)
            {
                ptr->value = newval;
                flag = 1;
                break;
            }
        }
        if (flag == 1)
        {
            printf("\nUpdated Successfully");
        }
        else
        {
            printf("\nValue not found in List");
        }
    }
}

/*
 * searching an element in a non-empty list
 */
void search()
{
    int flag = 0, key, pos = 0;

    if (first == NULL)
    {
        ISEMPTY;
        printf(":No nodes in the list\n");
    }
    else
    {
        printf("\nEnter the value to search");
        scanf("%d", &key);
        for (ptr = first;ptr != NULL;ptr = ptr->next)
        {
            pos = pos + 1;
            if (ptr->value == key)
            {
                flag = 1;
                break;
            }
        }
        if (flag == 1)
        {
            printf("\nElement  %d  found  at  %d  position\n",  key,
pos);
        }
        else
        {
            printf("\nElement %d not found in list\n", key);
        }
    }
}
/* Displays non-empty List from Beginning to End */
void display()
{
    if (first == NULL)
    {
        ISEMPTY;
        printf(":No nodes in the list to display\n");
    }
    else
    {
        for (ptr = first;ptr != NULL;ptr = ptr->next)
        {
            printf("%d\t", ptr->value);
        }
    }
}

/* Display non-empty list in Reverse Order */
void rev_display(snode *ptr)
{
    int val;

    if (ptr == NULL)
    {
        ISEMPTY;
        printf(":No nodes to display\n");
    }
    else
```

```
        {
            if (ptr != NULL)
            {
                val = ptr->value;
                rev_display(ptr->next);
                printf("%d\t", val);
            }
        }
}
```

Output:



```
1.Insert node at first
2.Insert node at last
3.Insert node at position
4.Sorted Linked List in Ascending Order
5.Delete Node from any Position
6.Update Node Value
7.Search Element in the linked list
8.Display List from Beginning to end
9.Display List from end using Recursion
10.Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice1

...Inserting node at first...

Enter the value for the node:10

----INSERTED----
YOU WANT TO CONTINUE (Y/N)y

---------------------------------

Operations on singly linked list

---------------------------------

1.Insert node at first
2.Insert node at last
3.Insert node at position
4.Sorted Linked List in Ascending Order
5.Delete Node from any Position
6.Update Node Value
7.Search Element in the linked list
8.Display List from Beginning to end
9.Display List from end using Recursion
10.Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice8

...Displaying List From Beginning to End...
10
YOU WANT TO CONTINUE (Y/N)n


------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.

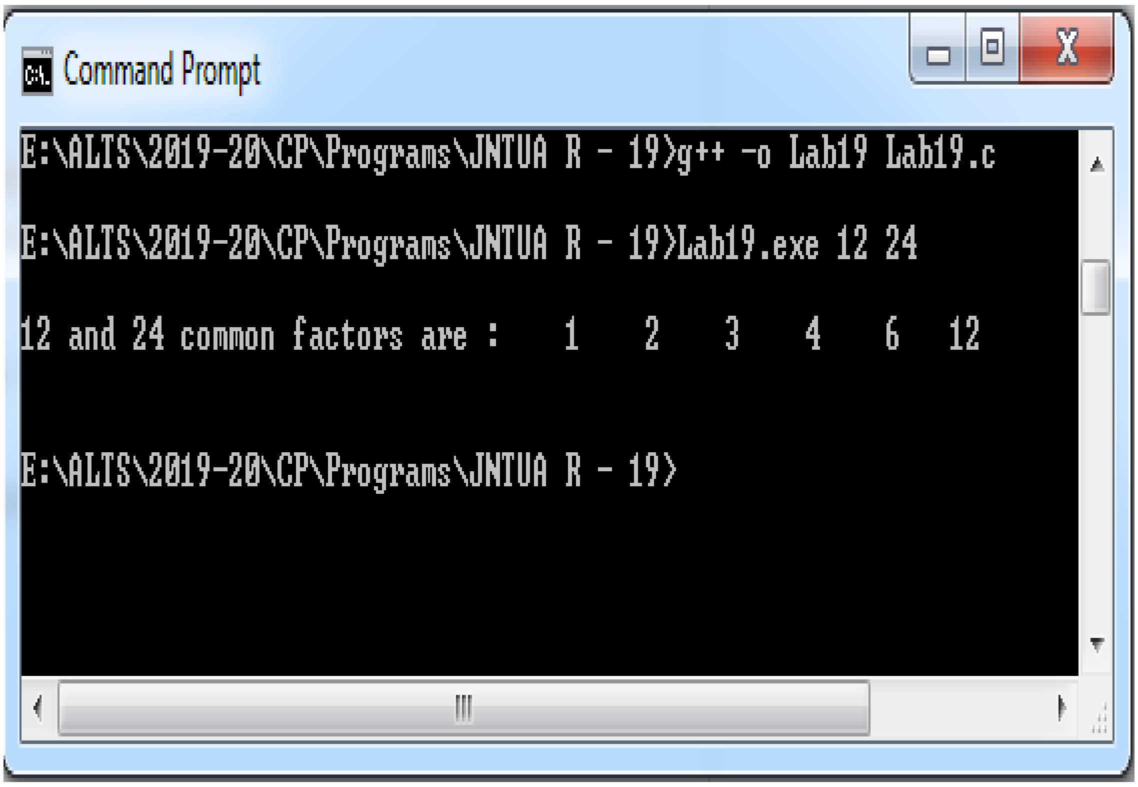
Week - 14:

Aim: Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.

Program:

```
#include<stdlib.h>
#include<stdio.h>

void commonfactors(int, int);

int main(int argc, char * argv[])
{
     int a, b;
     if (argc != 3)
     {
          printf("You have forgot to specify two numbers.");
          exit(1);
     }
     a = atoi(argv[1]);
     b = atoi(argv[2]);
     commonfactors(a, b);
     return 0;

}

void commonfactors(int num1, int num2)
{
     int i;
     printf("\n%d and %d common factors are :", num1, num2);
     for(i = 1; i <= num1; i++)
     {
          if (num1 % i == 0 && num2 % i == 0)
          {
               printf("%5d", i);
          }
     }
     printf("\n\n");
}
```

Output:



```
E:\ALTS\2019-20\CP\Programs\JNTUA R - 19>g++ -o Lab19 Lab19.c

E:\ALTS\2019-20\CP\Programs\JNTUA R - 19>Lab19.exe 12 24

12 and 24 common factors are :    1    2    3    4    6   12


E:\ALTS\2019-20\CP\Programs\JNTUA R - 19>
```

Result: The above program successfully executed.

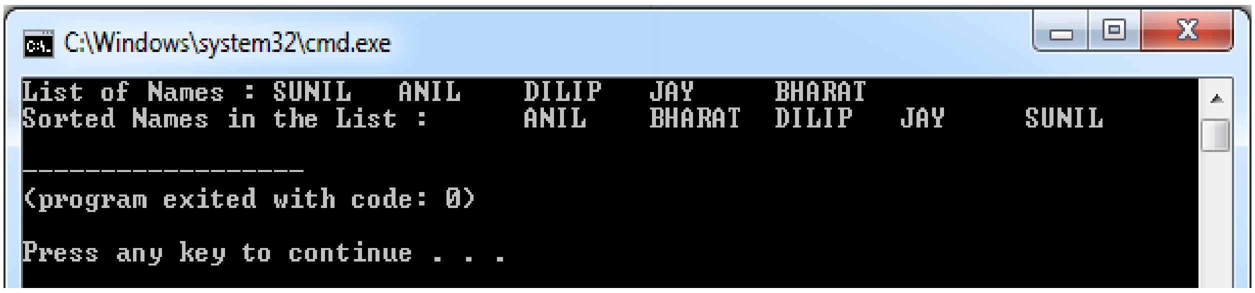
Week – 15:

Aim: Design a C program which sorts the strings using array of pointers.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
	char *t; int i,j,k;
	char *array[5]={"SUNIL","ANIL","DILIP","JAY","BHARAT"};
	printf("List of Names : ");
	for(i=0;i<5;i++)
		printf("%s \t",array[i]);
	for(i=0;i<4;i++)
	{
		for(j=0;j<4-i;j++)
		{
			k = strcmp(array[j],array[j+1]);
			if(k>0)
			{
				t=array[j];
				array[i]=array[j+1];
				array[j+1]=t;
			}
		}
	}
	printf("\nSorted Names in the List : ");
	for(i=0;i<5;i++)
		printf("\t%s",array[i]);
}
```

Output:



```
List of Names : SUNIL   ANIL    DILIP   JAY     BHARAT
Sorted Names in the List :      ANIL    BHARAT  DILIP   JAY     SUNIL

------------------
(program exited with code: 0)

Press any key to continue . . .
```

Result: The above program successfully executed.