A Lab Manual

On

COMPUTER NETWORKS & OPERATING SYSTEMS

(II- B. Tech. – II– Semester)

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING

By

Mrs.S.SHILPA

(Asst. Professor, Dept. of CSE)

&

Mr.MAHEEDHAR

(Asst. Professor, Dept. of CSE)

&

Mrs.S.SARIKA

(Asst. Professor, Dept. of CSE)

&

Mr.KUNAL

(Asst. Professor, Dept. of CSE)

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

Kandlakoya(V), Medchal Road, Hyderabad – 501 401

Ph. No. 08418-222042, 22106 Fax No. 08418-222106

(2023-24)

COMPUTER NETWORKS & OPERATING SYSTEMS LAB

II-B.Tech.-II-Sem.	**L**	**T**	**P**	**C**
Subject Code :22CDPC45	**0**	**0**	2	**1**

Course Outcomes

- Make use of NS2/NS3 tools in computer networks
- Outline the concepts of network models and components
- Adapt various data link layer algorithms and protocols
- Illustrate various network layer algorithms and protocols
- Demonstrate various transport layer algorithms and protocols

List Of Experiments:

1. Implement the data link layer framing method using character stuffing and bit stuffing.
2. Implement CRC on a data set of characters using CRC -12/ CRC- 16 polynomial.
3. Implement Stop and Wait Protocol.
4. Implement Sliding Window Protocol.
5. Implement Dijkstra ‘s shortest path through a graph.
6. Obtain broadcast tree for given subnet of hosts.
7. Implement collision free protocol.
8. a) Study of Linux general purpose utilities (File handling, Process, Disk, Networking, Filters) b) Implement Linux commands i) CP ii) MV
9. a) Write a shell script to find factorial of a given integer. b) Write a C program to create a child process and allow parent to display ‘parent’ and child to display ‘child’. c) Write a C program in which a parent writes a message to a pipe and the child reads the message
10. Write C programs to simulate the following CPU scheduling algorithms a) FCFS b) Priority
11. Write C programs to simulate the following CPU scheduling algorithms a) SJF b) RR
12. Write C programs to simulate the following file allocation strategies a) Sequential b) Linked c) Indexed
13. Write C programs to simulate the following memory management techniques a) Paging b) Segmentation
14. Write C programs to simulate the following page replacement techniques: a) FIFO b) LRU c) Optimal

References:

CN & OS (Linux) Lab Manual, Department of CSE , CMRIT, Hyd.

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

Kandlakoya(V), Medchal Road, Hyderabad – 501 401

Ph. No. 08418-222042, 22106 Fax No. 08418-222106

CMR INSTITUTE OF TECHNOLOGY

Vision: To create world class technocrats for societal needs.
Mission: Achieve global quality technical education by assessing learning environment through

- Innovative Research & Development
- Eco-system for better Industry institute interaction
- Capacity building among stakeholders

Quality Policy: Strive for global professional excellence in pursuit of key-stakeholders

DEPARTMENT OF CSE

Vision: Develop competent software professionals, researchers and entrepreneurs to serve global society.
Mission: The department of Computer Science and Engineering (Data Science) is committed to

- create technocrats with proficiency in design and code for software development
- adapt contemporary technologies by lifelong learning and face challenges in IT and ITES sectors
- quench the thirst of knowledge in higher education, employment, R·&D and entrepreneurship

I. Programme Educational Objectives (PEOs): Engineering Graduates will

1. Pursue successful professional career in IT and IT-enabled sectors.

2. Pursue lifelong learning skills to solve complex problems through multidisciplinary-research.

3. Exhibits professionalism, ethics and inter-personal skills to develop leadership qualities.

II. Programme Outcomes (POs): Engineering Graduates will be able to

1. Apply mathematics, science, engineering fundamentals to solve complex engineering problems.

2. Identify, formulate and analyze complex engineering problems to reach substantiated conclusions.

3. Design and develop a component/system/process to solve complex societal engineering problems.

4. Design and conduct experiments to analyze, interpret and synthesize data for valid conclusions.

5. Create, select and apply modern tools, skills, resources to solve complex engineering problems.

6. Apply contextual engineering knowledge to solve societal issues.

7. Adapt modern engineering practices with environmental safety and sustainable development.

8. Apply professional code of ethics, responsibilities and norms in engineering practices.

9. Compete as an individual and/or as a leader in collaborative cross cultural teams.

10. Communicate effectively through technical reports, designs, documentations and presentations.

11. Endorse cognitive management skills to prepare project report using modern tools and finance.

12. Engage in independent and life-long learning in the broad context of technological changes.

III. Programme Specific Outcomes (PSOs): Engineering Graduates will be able to

1. Design and develop Computer-Based-Systems using Algorithms, Networks, Security, Gaming, Full Stack, DevOps, IoT, Cloud, Data Science and AI&ML.

2. Apply data analytics to solve real world problems. _____

COURSE OUTCOMES:

Course Outcomes	Course Outcome Statements
CO -1	Implement data link protocols
CO -2	Find shortest path using routing table
CO -3	Illustrate linux shell environment
CO -4	Interpret CPU scheduling algorithms and file allocation methords
CO -5	Experiment with page replacement and memory management

COURSE MAPPING WITH PEO'S,PO'S,PSO'S
(No correlation:0,Low:1,Medium:2,High:3)

Course Title	PEO1	PEO2	PEO3	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CN Lab						3		3				3					3	

MAPPING OF COURSE OUTCOMES WITH PEO'S,PO'S,PSO'S
(No correlation:0,Low:1,Medium:2,High:3)

Course outcomes	PEO1	PEO2	PEO3	Po1	Po2	Po3	Po4	Po5	Po6	Po7	Po8	Po9	Po10	Po11	Po12	PSO1	PSO2	PSO3
CO - 1						3		3				3					3	
CO – 2						3		3				3					3	
CO – 3						3		3				3					3	
CO – 4						3		3				3					3	
CO – 5						3		3				3					3	

LESSON PLAN:

Week No.	Name of the Program	No. of Lab sessions	Text Books	Mode of Assessment
1	Implement the data link layer framing method using character stuffing and bit stuffing.	1	T1,T2	Viva&Execution
2	Implement CRC on a data set of characters using CRC -12/ CRC- 16 polynomial.	1	T1,T2	Viva&Execution
3	Implement Stop and Wait Protocol.	1	T1,T2	Viva&Execution
4	Implement Sliding Window Protocol.	1	T1,T2	Viva&Execution
5	Implement Dijkstra 's shortest path through a graph.	1	T1,T2	Viva&Execution
6	Obtain broadcast tree for given subnet of hosts.		T1,T2	Viva&Execution
7	Implement collision free protocol.	1	T1,T2	Viva&Execution
8	a) Study of Linux general purpose utilities (File handling, Process, Disk, Networking, Filters) b) Implement Linux commands i) CP ii) MV	1	T1,T2	Viva&Execution
9	a) Write a shell script to find factorial of a given integer. b) Write a C program to create a child process and allow parent to display 'parent' and child to display 'child'. c) Write a C program in which a parent writes a message to a pipe and the child reads the message	1	T1,T2	Viva&Execution
10	Write C programs to simulate the following CPU scheduling algorithms a) FCFS b) Priority	1	T1,T2	Viva&Execution
11	Write C programs to simulate the following CPU scheduling algorithms a) SJF b) RR	1	T1,T2	Viva&Execution
12	Write C programs to simulate the following file allocation strategies a) Sequential b) Linked c) Indexed	1	T1,T2	Viva&Execution
13	Write C programs to simulate the following memory management techniques a) Paging b) Segmentation	1	T1,T2	Viva&Execution
14	Write C programs to simulate the following page replacement techniques: a) FIFO b) LRU c) Optimal	1	T1,T2	Viva&Execution

NAME OF THE EXPERIMENT: 1(A) Bit Stuffing.

AIM: Implement the data link layer framing methods such as Bit stuffing.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-**
RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

The new technique allows data frames to contain an arbitrary number if bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with special bit pattern , 01111110, called a flag byte. When ever the senders data link layer encounters five consecutive one's in the data, it automatically stuffs a 0 bit into the out going bit stream.

ALGORITHM:

Step 1: Read a string of characters.

Step 2: Count the number of characters in the given string.

Step 3 : Display the count.

Step 4: Indicate using special characters start and end of the frame.

Step 5: Whenever the start character repeats in the frame, insert another same character to distinguish the character inside the frame.

Step 6: Display the same.

Step 7: Convert the given string into its equivalent bits.

Step 8: Whenever consecutive 5 1-bits appear in a string insert a 0 bit at the end of 5 consecutive bits.

Step 9: Display the string.

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[15];
int i,j,k,n,c=0,pos=0;
printf("\n enter the no of bits");
scanf("%d",&n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
{
if(a[i]==1)
{
c++;
if(c==5)
{
pos=i+1;
c=0;
for(j=n;j>=pos;j--)
{
k=j+1;
a[k]=a[j];
}
a[pos]=0;
n=n+1;
}
}
else
c=0;
}
printf("\n data after stuffing");
printf("011111110");
for(i=0;i<n;i++)
{
printf("%d",a[i]) ;
}
printf("011111110");
getch();
}
```

OUTPUT:

```
enter the no of bits
2
0
0
data after stuffing
01111111000011111110
```

NAME OF THE EXPERIMENT: 1(B) Character Stuffing.

AIM: Implement the data link layer framing methods such as character stuffing.

HARDWARE REQUIREMENTS: Intel based Desktop PC **,** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY: The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called character stuffing .

ALGORITHM

Begin

Step 1: Initialize I and j as 0

Step 2: Declare n and pos as integer and a[20],b[50],ch as character

Step 3: read the string a

Step 4: find the length of the string n, i.e n-strlen(a)

Step 5: read the position, pos

Step 6: if pos > n then

Step 7: print invalid position and read again the position, pos

Step 8: end if

Step 9: read the character, ch

Step 10: Initialize the array b , b[0…5] as ’d’, ’l’, ’e’, ’s’ ,’t’ ,’x’

respectively

Step 11: j=6;

Step 12: Repeat step[(13to22) until i<n

Step 13: if i==pos-1 then

Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’respectively

Step 15: increment j by 7, i.e j=j+7

Step 16: end if

Step 17: if a[i]==’d’ and a[i+1]==’l’ and a[i+2]==’e’ then

Step 18: initialize array b, b[13…15]=’d’, ‘l’, ‘e’ respectively

Step 19: increment j by 3, i.e j=j+3

Step 20: end if

Step 21: b[j]=a[i]

Step 22: increment I and j;

Step 23: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’,‘e’ ,’e’,‘t’, ‘x’,‘\0’

respectively

Step 24: print frame after stuffing

Step 25: print b

End

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
   char a[100],b[100],c[100];
   int i,j=0,n,k=0;
   printf("Enter the data");
   gets(a);
   n=strlen(a);

   for(i=0;i<n;i++)
   {

      if((a[i] == 'd' && a[i+1]=='l' && a[i+2]=='e')||
(a[i] == 'e' && a[i+1]=='s' && a[i+2]=='c'))
      {
         c[k++] = 'e';
         c[k++] = 's';
         c[k++] = 'c';
      }
      c[k++] = a[i];
   }
```

```
    c[k++] = '\0';

    b[j++]='\0';
    printf("%s\n",b);
    printf("DLESTX");
    printf("%s",c);
     printf("DLEETX");
    return 0;
}
```

OUTPUT:

```
Enter the data cradle
DLESTXcraescdleDLEETX
Process returned 0 (0X0) execution time:7.784s
Press any key to continue.
```

VIVA-VOCE

1. What is bit stuffing? What is the use of bit stuffing?

Bit stuffing is the process of inserting non-information bits into data to break up bit patterns to affect the synchronous transmission of information. Bit stuffing is commonly used to bring bit streams up to a common transmission rate or to fill frames.

2. What is character stuffing? What is the use of character stuffing?

In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. Each frame starts with the ASCII character sequence DLE STX and ends with the sequence DLE ETX.(where DLE is Data Link Escape, STX is Start of TeXt and ETX is End of TeXt.) This method overcomes the drawbacks of the character count method. If the destination ever loses synchronization, it only has to look for DLE STX and DLE ETX characters.

3. By which special bit pattern the frame begins and ends?

Each frame begins and ends with a special bit pattern called a flag byte [01111110].

4. What are the functions of data link layer?

1)It Provides well-defined service interface to the network layer on source machine to the network layer on destination machine.

2) The source machine sends data in blocks called frames to the destination machine. The starting and ending of each frame should be recognised by the destination machine.
3) The source machine must not send data frames at a rate faster than the destination machine can accept them.

5. Name the delimiters for character stuffing?

A)Each frame starts with the ASCII character sequence DLE STX and ends with the sequence DLE ETX

6. Expand DLE STX and DLE ETX?

DLE is Data Link Escape, STX is Start of TeXt

DLE is Data Link Escape, ETX is End of TeXt.

AME OF THE EXPERIMENT: 2) Cyclic Redundancy Check.

AIM: Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

HARDWARE REQUIREMENTS: Intel based Desktop PC**,**RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY: CRC method can detect a single burst of length n, since only one bit per column will be changed, a burst of length n+1 will pass undetected, if the first bit is inverted, the last bit is inverted and all other bits are correct. If the block is badly garbled by a long burst or by multiple shorter burst, the probability that any of the n columns will have the correct parity that is 0.5. so the probability of a bad block being expected when it should not be 2 power(-n). This scheme some times known as Cyclic Redundancy Code.

ALGORITHM

Implementation of CRC – 12

Step 1: Read values for transmitted data

Step 2: Compute polynomial for CRC-12

Step 3: Concatenate twelve 0's at the end of transmitted data

Step 4: Perform binary division where CRC-12 generating polynomial is the divisor and dividend is the concatenated string.

Step 5: Repeat the step 4 until length of remainder = = 12

Step 6: If remainder is less then 12, then add required number of zero's to the beginning of the remainder such that length is 12.

Step 7: Read received data

Step 8: Append CRC with received data

Step 9: Perform binary division on this with the divisor as generating polynomial

Step10: Repeat step 9 until remainder= =0

Step 11: If remainder= =0 then print message " data transmitted correctly"

Step12: Else print message “data transmitted incorrectly”

Implementation of CRC – 16

Step 1: Read values for transmitted data

Step 2: Compute polynomial for CRC-16

Step 3: Concatenate sixteen 0’s at the end of transmitted data

Step 4: Perform binary division where CRC-16 generating polynomial is the divisor and dividend is the concatenated string.

Step 5: Repeat the step 4 until length of remainder = = 16

Step 6: If remainder is less then 16, then add required number of zero’s to the beginning of the remainder such that length is 16.

Step 7: Read received data

Step 8: Append CRC with received data

Step 9: Perform binary division on this with the divisor as generating polynomial

Step10: Repeat step 9 until remainder= =0

Step 11: If remainder= =0 then print message “ data transmitted correctly”

Step12: Else print message “data transmitted incorrectly”

SOURCE CODE:

```
#include <stdio.h>
 #include <conio.h>
 #include <string.h>
 void main() {
        int i,j,keylen,msglen;
        char input[100], key[30],temp[30],quot[100],rem[30],key1[30];

        printf("Enter Data: ");
        gets(input);
        printf("Enter Key: ");
        gets(key);
        keylen=strlen(key);
        msglen=strlen(input);
        strcpy(key1,key);
        for (i=0;i<keylen-1;i++) {
                input[msglen+i]='0';
        }
        for (i=0;i<keylen;i++)
         temp[i]=input[i];
        for (i=0;i<msglen;i++) {
                quot[i]=temp[0];
                if(quot[i]=='0')
                 for (j=0;j<keylen;j++)
                 key[j]='0'; else
                 for (j=0;j<keylen;j++)
                 key[j]=key1[j];
                for (j=keylen-1;j>0;j--) {
                        if(temp[j]==key[j])
                         rem[j-1]='0'; else
                         rem[j-1]='1';
                }
                rem[keylen-1]=input[i+keylen];
```

```
                strcpy(temp,rem);
        }
        strcpy(rem,temp);
        printf("\nQuotient is ");
        for (i=0;i<msglen;i++)
         printf("%c",quot[i]);
        printf("\nRemainder is ");
        for (i=0;i<keylen-1;i++)
         printf("%c",rem[i]);
        printf("\nFinal data is: ");
        for (i=0;i<msglen;i++)
         printf("%c",input[i]);
        for (i=0;i<keylen-1;i++)
         printf("%c",rem[i]);
        getch();
}
```

OUTPUT:

```
Enter data:110001100011
Enter key:1101
Quotient is 100110011111
Remainder is:011
Final data is:110001100011011
```

VIVA-VOCE

1. **What is CRC?**
   A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption. CRCs can be used for error correction .

2. **What is the use of CRC?**
   The use of cyclic codes, which encode messages by adding a fixed-length check value,for the purpose of error detection in communication networks.

3. **Name the CRC standards for generator polynomial?**

   CRC-12: $x^{12}+x^{11}+x^3+x^2+x+1$

   CRC- 16: $X^{16}+x^{15}+x^2+1$

   CRC-CCITT: $x^{16}+x^{12}+x^5+1$

4. **How do you convert generator polynomial into binary form?**
   The generator polynomial is converted into the binary form by considering coefficients of polynomial
   EX: $x^{12}+x^{11}+x^3+x^2+x^1+1$ = 1100000001111

5. **Define checksum?**
   A checksum is a simple type of redundancy check that is used to detect errors in data

6. **How do you perform binary division operation in CRC?**

```
            10101100
        --------------
11011 ) 111001010000
          11011
          -----
           01111010000
            11011
            -----
             010110000
              11011
              -----
               1101000
               11011
               -----
                000100
```

NAME OF THE EXPERIMENT: 3. Stop and Wait Protocol

AIM: Implementation of Stop and Wait Protocol.

HARDWARE REQUIREMENTS: Intel based Desktop PC**,** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY: Stop-and-Wait Protocol means that the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming over-whelmed with frames, we need to tell the sender to slow down. A sliding window protocol is a feature of packet-based data transmission protocols. Conceptually, each portion of the transmission is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correct order.6. Sliding window protocol allows an unlimited number of packets to be communicated using fixed-size sequence numbers.7. The term "window" on transmitter side represents the logical boundary of the total number of packets yet to be acknowledged by the receiver.

ALGORITHM:

1. Start the program.
2. Create the socket by specifying the address and establishes the connection
3. Send and receive information.
4. The sender sends one frame, stops until it receives confirmation from the receiver and then sends the next frame.
5. Stop the program.

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int i,j,noframes,x,x1=10,x2;
for(i=0;i<200;i++)
rand();
noframes=rand()/200;
i=1;
j=1;
noframes=noframes/8;
printf("\n number of frames is %d",noframes);
while(noframes>0)
{
printf("\n sending frame %d",i);
srand(x1++);
x=rand()%10;
if(x%2==0)
{
for(x2=1;x2<2;x2++)
{printf("waiting for %d seconds\n",x2);
sleep(x2);
}
printf("\n sending frame %d",i);
srand(x1++);
x=rand()%10;
}
printf("\nack for frame %d",j);
noframes-=1;
```

```
i++;
j++;
}
printf("\nend of stop and wait protocol");
getch();
}
```

OUTPUT:

```
No.of Frames is 6
Sending Frame 1
Acknowledged for frame 1
Sending frame 2
Acknowledged for frame 2
Sending frame 3
Acknowledged for frame 3
Sending frame 4
Acknowledged for frame 4
Sending frame 5
Acknowledged for frame 5
Sending frame 6
Waiting for 1 second
Sending frame 6
Acknowledged for frame 6
End of stop and wait protocol
```

VIVA-VOCE:

1. **Explain Stop and Wait protocol?**
   The stop and wait protocol is a flow control protocol where flow control is one of the services of the data link layer It is a data-link layer protocol which is used for transmitting the data over the noiseless channels. It provides unidirectional data transmission which means that either sending or receiving of data will take place at a time. It provides flow-control mechanism but does not provide any error control mechanism.

2. **Explain the features of Stop and Wait protocol?**

   The features of Stop and Wait Protocol are as follows –

   - It is used in Connection-oriented communication.
   - It offers error and flows control.
   - It can be used in data Link and transport Layers.
   - Stop and Wait ARQ executes Sliding Window Protocol with Window Size

3. **What are the responsibilities of data link layer?**

   Specific responsibilities of data link layer include the following.

   - a) Framing
   - b) Physical addressing
   - c) Flow control
   - d) Error control
   - e) Access control

4. **Mention the categories of flow control.**

   There are 2 methods have been developed to control flow of data across communication links.

   - Stop and wait- send one from at a time.
   - Sliding window- send several frames at a time.

NAME OF THE EXPERIMENT: 4. Sliding Window Protocol

AIM: Implementation of Sliding Window Protocol

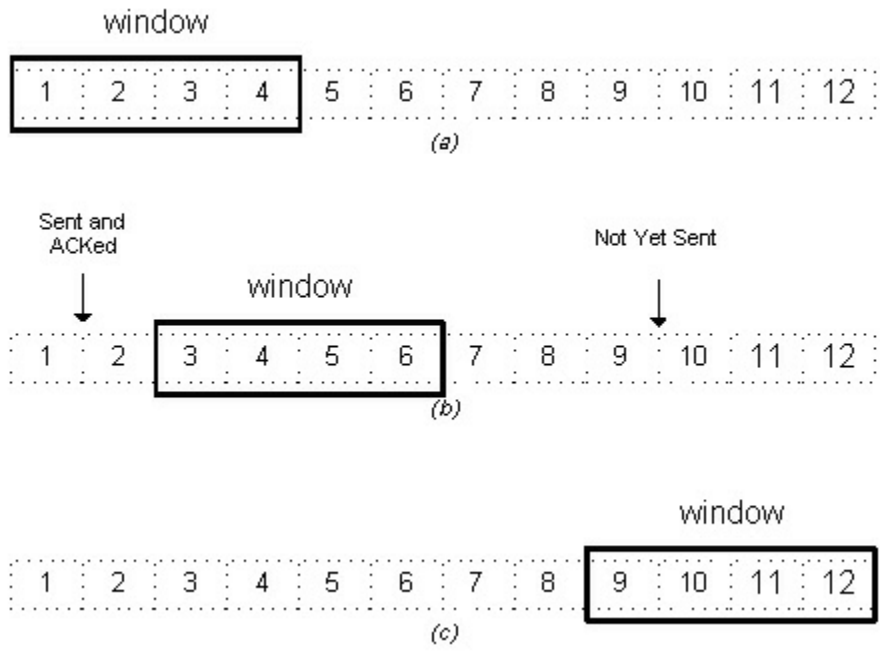
HARDWARE REQUIREMENTS: Intel based Desktop PC**,** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY: Sliding window protocol is a method to transmit data on a network. Sliding window protocol is applied on the Data Link Layer of OSI model. At data link layer data is in the form of frames. In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agree on some window size. If window size=w then after sending w frames sender waits for the acknowledgement (ack) of the first frame.

As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frames are properly received, for eg:- if ack of frame 3 is received it understands that frame 1 and frame 2 are received properly.



In sliding window protocol the receiver has to have some memory to compensate any loss in transmission or if the frames are received unordered.

Efficiency of Sliding Window Protocol

$\eta = (W*t_x)/(t_x+2t_p)$

W = Window Size

t_x = Transmission time

t_p = Propagation delay

Sliding window works in full duplex mode

It is of two types:-

1. Selective Repeat: Sender transmits only that frame which is erroneous or is lost.

2. Go back n: Sender transmits all frames present in the window that occurs after the error bit including error bit also.

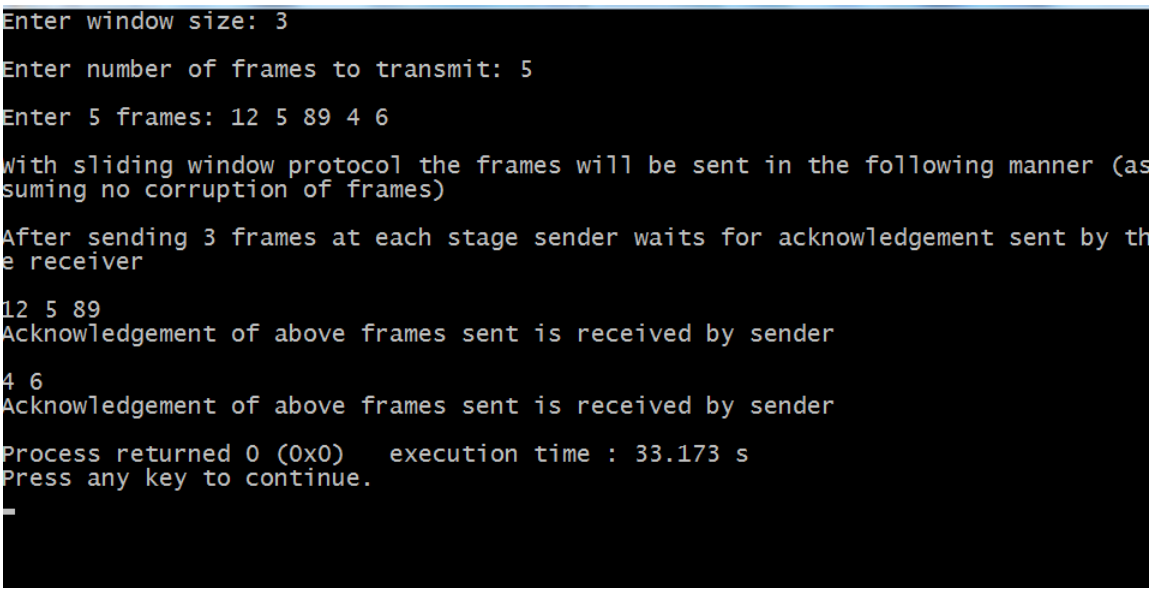
ALGORITHM:
Step 1: Start
Step 2: Read the size of the window.
Step 3: Select randomly the number of frames is to be transferred.
Step 4: Read the content of the input file.
Step 5: Transfer the frame until it reaches the maximum defined size.
Step 6: Resume the window size and repeat the above steps until frames are in.
Step 7: Stop

SOURCE CODE:

```
#include<stdio.h>
int main()
{
   int w,i,f,frames[50];

   printf("Enter window size: ");
   scanf("%d",&w);

   printf("\nEnter number of frames to transmit: ");
   scanf("%d",&f);

   printf("\nEnter %d frames: ",f);

   for(i=1;i<=f;i++)
      scanf("%d",&frames[i]);

   printf("\nWith sliding window protocol the frames will be sent in the
following manner (assuming no corruption of frames)\n\n");
   printf("After sending %d frames at each stage sender waits for
acknowledgement sent by the receiver\n\n",w);

   for(i=1;i<=f;i++)
   {
      if(i%w==0)
      {
         printf("%d\n",frames[i]);
         printf("Acknowledgement of above frames sent is received by
sender\n\n");
      }
      else
```

```
        printf("%d ",frames[i]);
    }

    if(f%w!=0)
        printf("\nAcknowledgement of above frames sent is received by
sender\n");

    return 0;
}
```

OUTPUT:



VIVA-VOCE:

1. **Explain sliding Window protocol?**

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI layer 2) as well as in the Transmission ControlProtocol (TCP).

2. **What are the different types of sliding window protocols?**

    1. 1.Stop and wait Protocol
    2. 2.GO-BACK-N (GBN) Protocol
    3. 3.Selective Repeat Protocol (SRP)

3. **Explain the benefits of sliding window protocol?**

    It controls the speed of transmission so that no fast sender can overwhelm the slower receiver;

    It allows for orderly delivery

    It allows for retransmission of lost frames, specific retransmission policy depends on the specific implementations.

4. **What are the functionalities of data link layer?**

    Data link layer deals with transmission errors

    1. 2. regulate the flow of data
    2. provide a well-defined interface to the network layer.

5. **Explain the difference between the sliding protocols?**

    Stop and wait –
    Sender window size (Ws) = 1
    Receiver window size (Wr) = 1
    Sequence Number >= 1 + 1
    Uses independent acknowledgement
    Discards out of order packets

    Packet Loss → Retransmit packet after time out

1. Acknowledgement loss → Resends packet after time out
2. Efficiency = 1/(1+2a) where a = Tp/Tt

Go Back N –

1. Sender window size Ws = N
2. Receiver window size Wr = 1
3. Sequence number >= N + 1
4. Can use both cumulative or independent acknowledgement depends on acknowledge timer
5. Discards out of order packets
6. Packet Loss → Track back N size from the last packet within the window limit to the lost packet and retransmit them
7. Acknowledgement loss → If not received before timeout the entire window N size is resend
8. Efficiency = N/(1+2a) where a = Tp/Tt

Selective Repeat –

1. Sender window size Ws = N
2. Receiver window size Wr = N
3. Sequence Number >= N + N
4. Uses only independent acknowledgement
5. Can Accept out of order packets
6. Packet Loss → Resend only the lost packet after timeout
7. Acknowledgement loss → Resend if not receive before timeout
8. Efficiency = N/(1+2a) where a = Tp/Tt

6. What is piggy backing? Why?

Piggybacking data is a bit different from Sliding Window Protocol **used** in the OSI model. In the data frame itself, we incorporate one additional field for acknowledgment (called ACK). ... If station A wants to send both data and an acknowledgment, it keeps both fields there.

NAME OF THE EXPERIMENT: 5. Shortest Path Algorithm.

AIM: Implement Dijkstra's algorithm to compute the Shortest path through a graph.

HARDWARE REQUIREMENTS: Intel based Desktop PC**,** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY: Build a graph of the subnet, with each node representing a router and each arc of the graph a communication link.

The labels on arc could be computed as a function of the delay. Initially, no paths are known, so all nodes are labeled with infinity, as algorithm proceeds and paths are found, labels may change, reflecting better paths. Initially all labels are tentative. When it is discovered, that a label represent the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

Each node is labeled with its distance from the source node along the best known path. This distance will be shortest from source to destination.

ALGORITHM:

Step 1: Plot the subnet, assign weights to each of the edges between nodes.

Step 2: Using the metrics as distance in km calculate the shortest path from the given source to destination.

Step 3: Initially mark all the paths from source as infinity.

Step 4: Starting with the source node check the adjacent nodes for shortest path, andmark them as tentative nodes.

Step 5: From these tentative nodes, select one which is having short distance from source, and mark as permanent.

Step 6: Distance from source to that tentative node should be recorded.

Step 7: Now this node is considered as source node and repeat the steps from 3 to 6.

Step 8: Repeat the steps along the path with the distances being added throughout the path to reach the destination.

SOURCE CODE:

```
#include <stdio.h>
#define infinity 9999
#define MAX 20
int minimum(int a,int b)
{
                        if(a<=b)
                          return a;
                        else
                          return b;
}
main()
{
                        int i,j,k,n,start,end,adj[MAX][MAX],path[MAX][MAX];
                        printf("Enter number of vertices : ");
                        scanf("%d",&n);
                        printf("Enter weighted matrix :\n");
                        for(i=0;i<n;i++)
                          for(j=0;j<n;j++)
                                scanf("%d",&adj[i][j]);
                            for(i=0;i<n;i++)
                          for(j=0;j<n;j++)
                          if(adj[i][j]==0)
                                path[i][j]=infinity;
                          else
                                path[i][j]=adj[i][j];
                        for(k=0;k<n;k++)
                        {
                                for(i=0;i<n;i++)
                                  for(j=0;j<n;j++){
          if(i==j)
          path[i][j]=infinity;
         else
path[i][j]=minimum(path[i][j],path[i][k]+path[k][j]);
                        }
```

```
                    }
                    printf("Shortest path matrix is :\n");
                    for(i=0;i<n;i++)
                    {
                        for(j=0;j<n;j++)
                            printf("%6d",path[i][j]);
                        printf("\n");
                    }
                    printf("Enter start vertex :");
                    scanf("%d",&start);
                    printf("Enter end vertex :");
                    scanf("%d",&end);
printf("the min. cost between %d and %d is %d",start,end,path[start][end]);
}
```

OUTPUT:

```
Enter number of vertices:4
Enter weighted matrix:
0 1 3 0
1 0 0 4
3 0 0 4
0 4 4 0
Shortest path matrix is
9999    1      3      5
1       9999   4      4
3       4      9999   4
5       4      4      9999
Enter start vertex:1
Enter end vertex:3
The min cost between 1 and 3 is 4
Process returned 34(0X22) execution time:69.346s
Press any key to continue.
```

VIVA VOCE:

1. **What is Flow based routing algorithm?**

Flow-based routing seeks to find a routing table to minimize the average packet delay through the subnet.

2. **What is the Link state routing algorithm?**

Link State improves the convergence of Distance Vector by having everybody share their idea of the state of the net with everybody else (more information is available to nodes, so better routing tables can be constructed).

3. **In shortest path which metric is considered?**

Shortest Path Metric is used in Shortest path Routing Algorithm

4. **What is the another name for shortest path algorithm?**

Another name for ShortestPath Algorithm is Dijkstra's algorithm

5. **What is the disadvantage of Dijkstra's algorithm?**

The major disadvantage of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources. Another disadvantage is that it cannot handle negative edges. This leads to acyclic graphs and most often cannot obtain the right shortest path.

6. **What is the advantage of Dijkstra's algorithm?**

1) It is used in Google Maps

2) It is used in finding Shortest Path.

3) It is used in geographical Maps

4) To find locations of Map which refers to vertices of graph.

5) Distance between the location refers to edges.

6) It is used in IP routing to find Open shortest Path First.

NAME OF THE EXPERIMENT: 6. Broadcast tree

AIM: Take an example subnet of hosts. Obtain broadcast tree for it.

HARDWARE REQUIREMENTS: Intel based Desktop PC**,** RAM of 512 MB
SOFTWARE REQUIREMENTS: Turbo C / Borland C.

Theory: Broad cast tree is one through which broad cast messages will be sent to all the nodes in the network. Broad cast tree is constructed using spanning tree.

Spanning tree algorithm is given below. After the construction of spanning tree, any node in the network can send broad cast message to all nodes in the network. For example, if node A sends an broad cast message to its nearby node B. then B will forward the packet on all of its outgoing lines except on which packet arrived. Using this method any node can send an broad cast packet in the network.

ALGORITHM:

Step 1: Pick any vertex as a starting vertex. (Call it S). Mark it with any given colour, say red.

Step 2: Find the nearest neighbor of S (call it P_1). Mark both P_1 and the edge SP_1 red. Cheapest unmarked (uncolored) edge in the graph that doesn't close a coloured circuit. Mark this edge with same colour of Step 1.

Step 3: Find the nearest uncoloured neighbour to the red subgraph (i.e., the closest vertex to any red vertex). Mark it and the edge connecting the vertex to the red subgraph in red.

Step 4: Repeat Step 2 until all vertices are marked red. The red subgraph is a minimum spanning tree

Step 5: Send an broad cast packet from node A to its near by nodes B and C.

Step 6: Nodes B and C should forward packet on its out going lines except on which packet arrived.

Step 7: Steps 5 and 6 are repeated until all nodes receive the packet.

SOURCE CODE:

```
#include<stdio.h>
int a[10][10],n;
main(){
int i,j,root;
printf("Enter no.of nodes:");
scanf("%d",&n);
printf("Enter adjacent matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++){
printf("Enter connecting of %d-->%d::",i,j);
scanf("%d",&a[i][j]);
}
printf("Enter root node:");
scanf("%d",&root);
adj(root);
}
adj(int k){
int i,j;
printf("Adjacent node of root node::\n");
printf("%d\n\n",k);
for(j=1;j<=n;j++)
{
if(a[k][j]==1 || a[j][k]==1)
printf("%d\t",j);
}
printf("\n");
for(i=1;i<=n;i++){
if((a[k][j]==0) && (a[i][k]==0) && (i!=k))
printf("%d",i);
}
}
```

OUTPUT:

```
Enter no.of nodes:5
Enter adjacent matrix
Enter connecting of 1-->1::0
Enter connecting of 1-->2::1
Enter connecting of 1-->3::1
Enter connecting of 1-->4::0
Enter connecting of 1-->5::0
Enter connecting of 2-->1::1
Enter connecting of 2-->2::0
Enter connecting of 2-->3::1
 Enter connecting of 2-->4::1
Enter connecting of 2-->5::0
Enter connecting of 3-->1::1
Enter connecting of 3-->2::1
Enter connecting of 3-->3::0
Enter connecting of 3-->4::0
Enter connecting of 3-->5::0
Enter connecting of 4-->1::0
Enter connecting of 4-->2::1
Enter connecting of 4-->3::0
Enter connecting of 4-->4::0
Enter connecting of 4-->5::1
Enter connecting of 5-->1::0
Enter connecting of 5-->2::0
Enter connecting of 5-->3::0
Enter connecting of 5-->4::1
Enter connecting of 5-->5::0
Enter root node:2
Adjacent node of root node::
2
1  3  4
 5
```

VIVA-VOCE:

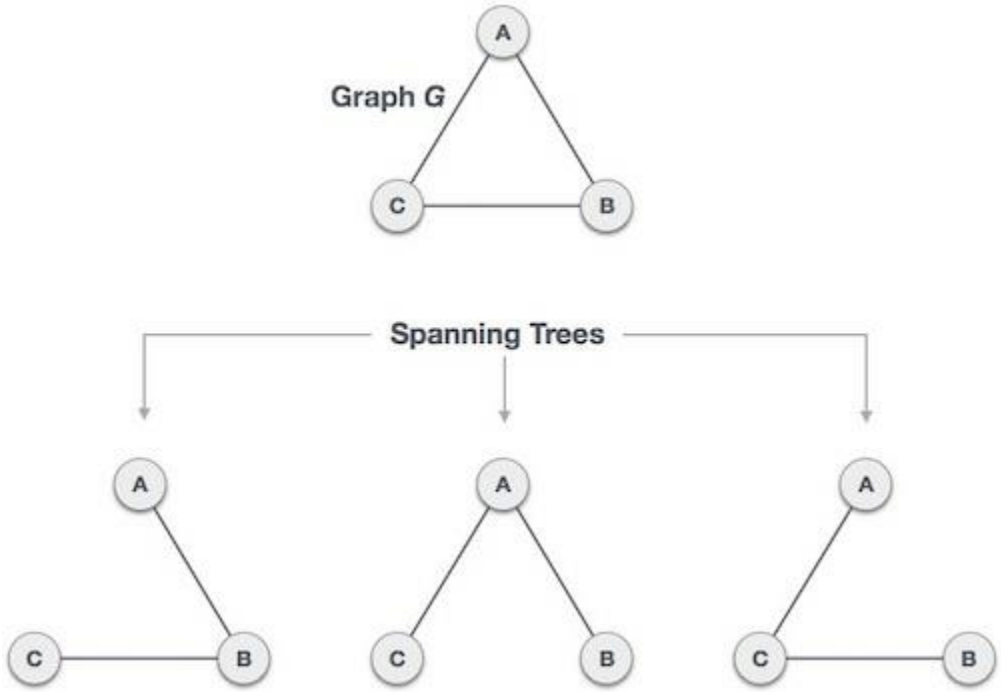
1. **What is spanning tree?**

Spanning Tree Protocol. The Spanning Tree Protocol (STP) is a network protocol that builds a loop-free logical topology for Ethernet networks

2.What is broadcasting?

Broadcast: Here, traffic streams from a single point to all possible endpoints within reach on the network, which is generally a LAN. This is the easiest technique to ensure traffic reaches to its destinations.

3.Design the spanning tree.



4.What is reverse path forwarding?

Reverse path forwarding. Reverse path forwarding (RPF) is a technique used in modern routers for the purposes of ensuring loop-free forwarding of multicast packets in multicast routing and to help prevent IP address spoofing in unicast routing.

5.How spanning tree helps to broadcast the message?

The need for the Spanning Tree Protocol (STP) arose because switches in local area networks (LANs) were often interconnected using redundant links to improve resilience should one connection, called a link, fail. However, this was found to create transmission loops, broadcast storms and MAC address table trashing. If redundant links are used to connect switches, then transmission loops need to be avoided[4] because data link layer 2 Ethernet frames do not expire. Potentially an Ethernet frame with a destination MAC address that is not in the MAC address table of the immediate switch can

be bounced around between switches in the local area network. Redundant links between these switches could result in the Ethernet frame never reaching a Switch that has the destination MAC address in its MAC address table. In such cases switches also broadcast the Ethernet frames to all ports, except the one from which it entered. This can create a broadcast storm

6. **What is the difference between Unicast, multicast and Broadcast?**

Unicast: traffic, many streams of IP packets that move across networks flow from a single point, such as a website server, to a single endpoint such as a client PC. This is the most common form of information transference on networks.

Broadcast: Here, traffic streams from a single point to all possible endpoints within reach on the network, which is generally a LAN. This is the easiest technique to ensure traffic reaches to its destinations.

Multicast: In this method traffic recline between the boundaries of unicast (one point to one destination) and broadcast (one point to all destinations). And multicast is a "one source to many destinations" way of traffic distribution, means that only the destinations that openly point to their requisite to accept the data from a specific source to receive the traffic stream.

NAME OF THE EXPERIMENT: 7. Implement collision free protocol

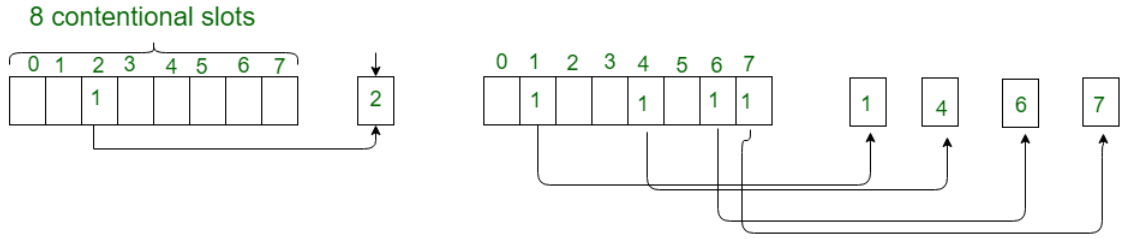
AIM: Implementation of Bit Map Protocol.

HARDWARE REQUIREMENTS: Intel based Desktop PC**,** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY: Bit map protocol is collision free Protocol. In bitmap protocol method, each contention period consists of exactly N slots. If any station has to send frame, then it transmits a 1 bit in the corresponding slot. For example, if station 2 has a frame to send, it transmits a 1 bit to the 2nd slot.

In general, Station 1 Announce the fact that it has a frame questions by inserting a 1 bit into slot 1. In this way, each station has complete knowledge of which station wishes to transmit. There will never be any collisions because everyone agrees on who goes next. Protocols like this in which the desire to transmit is broadcasting for the actual transmission are called Reservation Protocols.



A Bit-map Protocol.

For analyzing the performance of this protocol, We will measure time in units of the contention bits slot, with a data frame consisting of d time units. Under low load conditions, the bitmap will simply be repeated over and over, for lack of data frames. All the stations have something to send all the time at high load, the N bit contention period is prorated over N frames, yielding an overhead of only 1 bit per frame.

Generally, high numbered stations have to wait for half a scan before starting to transmit low numbered stations have to wait for half a scan(N/2 bit slots) before starting to transmit, low numbered stations have to wait on an average 1.5 N slots.

ALGORITHM:

Step 1 – We use the bitmap or the bit vector which represents a finite set of distinct integers.

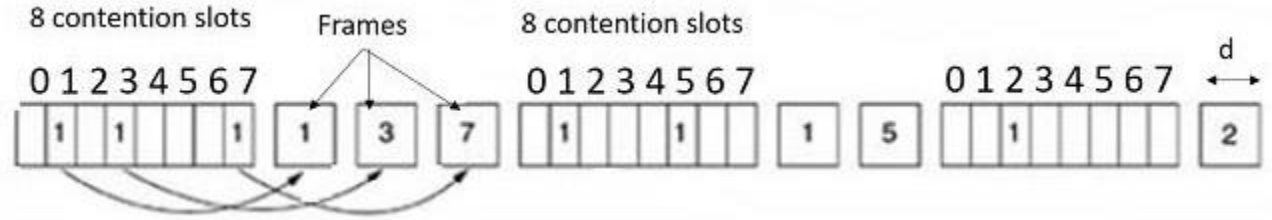
Step 2 – To sort the array of integers, initialization of the array size to the specified range is needed and then fill it with zeroes which is a default value

in the program followed by setting the corresponding bit in the bitmap to 1 for each integer that was input.

Step 3 – Scanning the bitmap and printing the integers in a sorted order being the final step.

Explanation

The Bit Map protocol is diagrammatically represented as follows –



Here,

Step 1 – Each contention period has exactly N slots. If a station 0 has a frame to send then it transmits 1 bit during slot 0. In general station j may announce that it has a frame to send by inserting 1 bit into slot j.

Step 2 – After all N slots have passed, then each station gets an idea which station is ready to transmit, then the frames are transmitted in numerical order.

Step 3 – Because of mutual understanding there is no chance of collision.

Step 4 – After the last ready station is transmitted its frame, all stations can monitor, another N-bit contention period begins.

Step 5 – If a station becomes ready just after its bit slots have passed by, it must remain silent until the bitmap has come around again.

Step 6 – Protocols like this in which the desire to transmit is broadcast before the actual transmission are called as reservation protocols because they reserve channel ownership in advance and prevent collisions.

SOURCE CODE:

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>

void main()
{

int i,j,k;

int sn;
int st[20];
cout<<"\n How many stations: ";
cin>>sn;
char op;
do
{
cout<<"\n Enter status of stations";
for (i=0;i<sn;i++)
{
   cout<<"\n Enter status of station "<<i+1<<" : ";
   cin>>st[i];
}
//Print ready stations
for (i=0;i<sn;i++)
{
   if(st[i]==1)
   {
      cout<<"\n Station "<<i+1<<" is ready to transmit";
   }
}
cout<<"\n Repeat? Press Y :";
cin>>op;
}
while(op=='y' || op=='Y');
getch();
}
```

OUTPUT:

```
How many stations: 8

 Enter status of stations
 Enter status of station 1 : 0

 Enter status of station 2 : 1

 Enter status of station 3 : 0

 Enter status of station 4 : 1

 Enter status of station 5 : 1

 Enter status of station 6 : 1

 Enter status of station 7 : 0

 Enter status of station 8 : 0

 Station 2 is ready to transmit
 Station 4 is ready to transmit
 Station 5 is ready to transmit
 Station 6 is ready to transmit
 Repeat? Press Y :Y
  Enter status of stations
 Enter status of station 1 : 1

 Enter status of station 2 : 0

 Enter status of station 3 : 0

 Enter status of station 4 : 1

 Enter status of station 5 : 0

 Enter status of station 6 : 0
 Enter status of station 7 : 0

 Enter status of station 8 : 0

 Station 1 is ready to transmit
 Station 4 is ready to transmit
 Repeat? Press Y :N
```

VIVA –VOCE:

1. Explain ARP

Address Resolution Protocol (**ARP**) is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network. For example, in IP Version 4, the most common level of IP in use today, an address is 32 bits long.

2. Explain the purpose of Network layer?

The network layer provides the means of transferring variable-length network packets from a source to a destination host via one or more networks. ... Host addressing. Every host in the network must have a unique address that determines where it is.

3. What Is The Use Of Arp?

A host in an Ethernet network can communicate with another host, only if it knows the Ethernet address (MAC address) of that host. The higher level protocols like IP use a different kind of addressing scheme (like IP address) from the lower level hardware addressing scheme like MAC address. ARP is used to get the Ethernet address of a host from its IP address. ARP is extensively used by all the hosts in an Ethernet network.

4. To Which Osi Layer Does Arp Belong?

ARP belongs to the OSI data link layer (Layer 2). ARP protocol is implemented by the network protocol driver. ARP packets are encapsulated by Ethernet headers and transmitted.

5. Explain RARP?

The Reverse Address Resolution Protocol (**RARP**) is an obsolete computer networking protocol used by a client computer to request its Internet Protocol (IPv4) address from a computer network, when all it has available is its link layer or hardware address, such as a MAC address.

6. What is the difference between ARP and RARP?

Address Resolution Protocol is utilized for mapping IP network address to the hardware address that uses data link protocol.

Reverse Address Resolution Protocol is a protocol using which a physical machine in a LAN could request to find its IP address from ARP table or cache from a gateway server.

WEEK 8

a)NAME OF THE EXPERIMENT: Implement Linux general purpose utilities

AIM: Study of Linux general purpose utilities(file handling, process utilities, disk utilities, networking and filters).

ALGORITHM:

Step 1: Start

Step 2: open Terminal

Step 3:give the proper command and check with the different type of the options

Step 4:observe the result of the each options

Step 5: stop

SOURCE CODE

File handling utilities:

Mkdir :make directories

Usage: mkdir [OPTION] DIRECTORY...

eg. mkdir laxmi

ls : list directory contents

Usage: ls [OPTION]... [FILE]...

eg. ls, ls l,

ls laxmi

cd : changes directories

Usage: cd [DIRECTORY]

eg. cd laxmi

pwd: print name of current working directory

Usage: pwd

vim : Vi Improved, a programmers text editor

Usage: vim [OPTION] [file]...

eg. vim file1.txt

cp :copy files and directories

Usage: cp [OPTION]... SOURCE DEST

eg. cp sample.txt sample_copy.txt

cp :sample_copy.txt target_dir

mv : move (rename) files

Usage: mv [OPTION]... SOURCE DEST

eg. mv source.txt target_dir

mv old.txt new.txt

rm: remove files or directories

Usage: rm [OPTION]... FILE...

eg. rm file1.txt , rm rf

some_dir

find:search for files in a directory hierarchy

Usage: find [OPTION] [path] [pattern]

eg. find file1.txt, find name

file1.txt

history:prints recently used commands

Usage: history

Security filespermissions:

3 types of file permissions – read, write, execute

• 10 bit format from 'ls l'

command

1 2 3 4 5 6 7 8 9 10

file type owner group others

eg. drwxrw-r—means owner has all three permissions,

group has read and write, others have only read permission

• **read permission – 4, write – 2, execute -1**

eg.rwxrw-r-- = 764

673 = rw-rwx-wx

chmod:change file access permissions

Usage: chmod [OPTION] [MODE] [FILE]

eg. chmod 744 calculate.sh

chown: change file owner and group

Usage: chown [OPTION]... OWNER[:[GROUP]] FILE...

eg. chown remo myfile.txt

su : change user ID or become superuser

Usage: su [OPTION] [LOGIN]

eg. su remo, su

passwd : update a user's authentication tokens(s)

Usage: passwd [OPTION]

eg. passwd

who: show who is logged on

Usage: who [OPTION]

eg. who , who b , who q

Process utilities:

ps : report a snapshot of the current processes

Usage: ps [OPTION]

eg. ps, ps el

kill :to kill a process(using signal mechanism)

Usage: kill [OPTION] pid

eg. kill 9

2275

Tar: to archive a file

Usage: tar [OPTION] DEST SOURCE

eg. tar cvf

/home/archive.tar /home/original

tar xvf

/home/archive.tar

Zip:package and compress (archive) files

Usage: zip [OPTION] DEST SOURSE

eg. zip original.zip original

unzip: list, test and extract compressed files in a ZIP archive

Usage: unzip filename

eg. unzip original.zip

Disk utilities:

du (abbreviated from *disk usage*) is a standard Unixprogram used to estimate file space usage—space used under a particular directory or files on a file system**.**

du takes a single argument, specifying a pathname for du to work; if it is not specified, the current directory is used. The SUS mandates for du the following options:

-a, display an entry for each file (and not directory) contained in the current directory

-H, calculate disk usage for link references specified on the command line

-k, show sizes as multiples of 1024 bytes, not 512-byte

-L, calculate disk usage for link references anywhere

-s, report only the sum of the usage in the current directory, not for each file

-x, only traverse files and directories on the device on which the pathname argument is specified.

Other Unix and Unix-like operating systems may add extra options. For example, BSD and GNU du specify a -h option, displaying disk usage in a format easier to read by the user, adding units with the appropriate SI prefix'

$ du-sk*

152304 directoryOne

1856548 directoryTwo

Sum of directories in human-readable format (Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte):

$ du-sh*

149M directoryOne

1.8G directoryTwo

disk usage of all subdirectories and files including hidden files within the current directory (sorted by filesize) :

$ du-sk .[!.]**|sort-n

disk usage of all subdirectories and files including hidden files within the current directory (sorted by reverse filesize) :

$ du-sk .[!.]**|sort–nr

The weight of directories:

$ du-d1-c-h

df command : Report file system disk space usage

Df command examples - to check free disk space

Typedf -h or df -k to list free disk space:
$ df -h
OR
$ df –k

Output:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdb1	20G	9.2G	9.6G	49%	/
varrun	393M	144k	393M	1%	/var/run
varlock	393M	0	393M	0%	/var/lock
procbususb	393M	123k	393M	1%	/proc/bus/usb
udev	393M	123k	393M	1%	/dev
devshm	393M	0	393M	0%	/dev/shm
lrm	393M	35M	359M	9%	/lib/modules/2.6.20-15-generic/volatile
/dev/sdb5	29G	5.4G	22G	20%	/media/docs
/dev/sdb3	30G	5.9G	23G	21%	/media/isomp3s
/dev/sda1	8.5G	4.3G	4.3G	51%	/media/xp1
/dev/sda2	12G	6.5G	5.2G	56%	/media/xp2
/dev/sdc1	40G	3.1G	35G	9%	/media/backup

du command examples

du shows how much space one ore more files or directories is using.
$ du -sh
103M

-s option summarize the space a directory is using and -h option provides "Human-readable" output.

Networking commands:

These are most useful commands in my list while working on Linux server , this enables you to quickly troubleshoot connection issues e.g. whether other system is connected or not , whether other host is responding or not and while working for FIX connectivity for advanced trading system this tools saves quite a lot of time .

• finding host/domain name and IP address - **hostname**
• test network connection – **ping**
• getting network configuration – **ifconfig**
• Network connections, routing tables, interface statistics – **netstat**
• query DNS lookup name – **nslookup**
• communicate with other hostname – **telnet**
• outing steps that packets take to get to network host – **traceroute**
• view user information – **finger**
• checking status of destination host - **telnet**

Example of Networking commands in Unix

let's see some example of various networking command in Unix and Linux. Some of them are quite basic e.g. ping and telnet and some are more powerful e.g. nslookup and netstat. When you used these commands in combination of find and grep you can get anything you are looking for e.g. hostname, connection end points, connection status etc.

hostname

hostnamewith no options displays the machines host name
hostname –ddisplays the domain name the machine belongs to
hostname –fdisplays the fully qualified host and domain name
hostname –idisplays the IP address for the current machine

ping
It sends packets of information to the user-defined source. If the packets are received, the

destination device sends packets back. Ping can be used for two purposes

1. To ensure that a network connection can be established.
2. Timing information as to the speed of the connection.

If you **do ping www.yahoo.com** it will display its IP address. Use ctrl+C to stop the test.

ifconfig

View network configuration, it displays the current network adapter configuration. It is handy to determine if you are getting transmit (TX) or receive (RX) errors.

netstat

Most useful and very versatile for finding connection to and from the host. You can find out all the multicast groups (network) subscribed by this host by issuing **"netstat -g"**

netstat -nap | grep portwill display process id of application which is using that port
netstat -a or netstat –allwill display all connections including TCP and UDP
netstat --tcp or netstat –twill display only TCP connection
netstat --udp or netstat –uwill display only UDP connection
netstat -gwill display all multicast network subscribed by this host.

nslookup

If you know the IP address it will display hostname. To find all the IP addresses for a given domain name, the command nslookup is used. You must have a connection to the internet for this utility to be useful.
E.g. **nslookup blogger.com**

You can also use nslookup to convert hostname to IP Address and from IP Address from hostname.

traceroute

A handy utility to view the number of hops and response time to get to a remote system or web site is traceroute. Again you need an internet connection to make use of this tool.

finger

View user information, displays a user's login name, real name, terminal name and write status. this is pretty old unix command and rarely used now days.

telnet

Connects destination host via telnet protocol, if telnet connection establish on any port means connectivity between two hosts is working fine.

telnet hostname port will telnet hostname with the port specified. Normally it is used to see whether host is alive and network connection is fine or not.

10 Most important linux networking commands

Linux is most powerful operating system which often needs to use commands to explore it effectively.Some of the commands are restricted to normal user groups as they are powerful and has more functionality involved in it.Here we summarized most interesting and useful networking commands which every linux user are supposed to be familiar with it.

1.Arp manipulates the kernel's ARP cache in various ways. The primary options are clearing an address mapping entry and manually setting up one. For debugging purposes, the arp program also allows a complete dump of the ARP cache.ARP displays the IP address assigned to particular ETH card and mac address

```
[fasil@smashtech]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
59.36.13.1               ether   C                   eth0
```

2.Ifconfig is used to configure the network interfaces. Normally we use this command to check the IP address assigned to the system.It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

```
[fasil@smashtech~]#/sbin/ifconfig
eth0      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:126341 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44441 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

3. Netstat prints information about the networking subsystem. The type of information which is usually printed by netstat are Print network connections, routing tables, interface statistics, masquerade connections, and multicast.

```
[fasil@smashtech ~]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
```

```
tcp        0      0                .230.87:https           ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  10     [ ]         DGRAM                    4970   /dev/log
unix  2      [ ]         DGRAM                    6625   @/var/run/hal/hotplug_socket
unix  2      [ ]         DGRAM                    2952   @udevd
unix  2      [ ]         DGRAM                    100564
unix  3      [ ]         STREAM     CONNECTED     62438  /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     62437
unix  3      [ ]         STREAM     CONNECTED     10271  @/tmp/fam-root-
unix  3      [ ]         STREAM     CONNECTED     10270
unix  3      [ ]         STREAM     CONNECTED     9276
unix  3      [ ]         STREAM     CONNECTED     9275
```

4.ping command is used to check the connectivity of a system to a network.Whenever there is problem in network connectivity we use ping to ensure the system is connected to network.

```
[root@smashtech ~]# ping google.com
PING google.com (74.125.45.100) 56(84) bytes of data.
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=0 ttl=241 time=295 ms
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=1 ttl=241 time=277 ms
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=2 ttl=241 time=277 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6332ms
rtt min/avg/max/mdev = 277.041/283.387/295.903/8.860 ms, pipe 2
```

5.Nslookup is a program to query Internet domain name servers. Nslookup has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

```
[fasil@smashtech ~]# nslookup google.com
Server:         server ip
Address:        gateway ip 3

Non-authoritative answer:
Name:   google.com
Address: 209.85.171.100
Name:   google.com
Address: 74.125.45.100
Name:   google.com
Address: 74.125.67.100
```

6. dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than dig.

```
[fasil@smashtech ~]# dig google.com

; <<>> DiG 9.2.4 <<>> google.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4716
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.            IN      A

;; ANSWER SECTION:
google.com.            122     IN      A       74.125.45.100
google.com.            122     IN      A       74.125.67.100
google.com.            122     IN      A       209.85.171.100

;; AUTHORITY SECTION:
google.com.            326567  IN      NS      ns3.google.com.
google.com.            326567  IN      NS      ns4.google.com.
google.com.            326567  IN      NS      ns1.google.com.
google.com.            326567  IN      NS      ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.        152216  IN      A       216.239.32.10
ns2.google.com.        152216  IN      A       216.239.34.10
ns3.google.com.        152216  IN      A       216.239.36.10
ns4.google.com.        152216  IN      A       216.239.38.10

;; Query time: 92 msec
;; SERVER: 172.29.36.1#53(172.29.36.1)
;; WHEN: Thu Mar  5 14:38:45 2009
;; MSG SIZE  rcvd: 212
```

7.Route manipulates the IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the ifconfig program.When the add or del options are used, route modifies the routing tables. Without these options, route displays the current contents of the routing tables.

```
[fasil@smashtech ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
54.192.56.321   *               255.255.255.0   U     0      0        0 eth0
        *       255.255.0.0     U     0      0        0 eth0
default         0.0.0.0         UG    0      0        0 eth0
```

8.Traceroute : Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking the route one's packets follow (or finding the miscreant gateway that's discarding your packets) can be difficult.

Traceroute utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host. The only mandatory parameter is the destination host name or IP number. The default probe datagram length is 40 bytes, but this may be increased by specifying a packet length (in bytes) after the destination host name.

```
[fasil@smashtech ~]# traceroute google.com
traceroute: Warning: google.com has multiple addresses; using 209.85.171.100
traceroute to google.com (209.85.171.100), 30 hops max, 38 byte packets
 1  * * *
```

9.W-displays information about the users currently on the machine, and their processes. The header shows, in this order, the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

```
[fasil@smashtechl ~]# w
 15:18:22 up  4:38,  3 users,  load average: 0.89, 0.34, 0.19
USER   TTY     FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
root   :0      -               10:41   ?xdm?  24:53   1.35s /usr/bin/gnome-session
root   pts/1   :0.0            10:58    1.00s  0.34s  0.00s w
root   pts/2   :0.0            12:10   23:32   0.03s  0.03s bash
```

Filters:

Filters are commands which accept data from standard input, manupulate it and write the results to standard output

Head: command displays the top of the file, when used without any option it will display first 10 lines of the file

```
$ head sample1.txt

/*display first 10 lines*/
```

tail:command displays the end of the file. By default it will display last 10 lines of the

file

$ tail sample1.txt

/*display last 10 lines*/

tail or head with -n followed by a number will display that many number of lines from last and from first respectively

$ head -n 20 sample1.txt

/* will display first 20 lines*/

$ tail -n 15 sample1.txt

/* will display last 15 lines */

cut : cutting columns

cut command can be used to cut the columns from a file with -c option

usage: cut -c [numbers delemited by comma or range]

<file name>

$ cut -c 1,2,3-5 students.txt

1 ani

2 das

3 shu

4 sin

cut : cutting fields

With -f option you can cut the feilds delemited by some character

$ cut -d" " -f1,4 students.txt

1 Mtech

2 Btech

3 Mtech

-d option is used to specify the delimiter and -f option used to specify the feild number

paste : pasting side by side

paste command will paste the contents of the file side by side

$ paste cutlist1.txt cutlist2.txt

1 Mtech 1 anil H1

2 Btech 2 dasgupta H4

3 Mtech 3 shukla H7

4 Mtech 4 singhvi H12

5 Btech 5 sumit H13

sort : ordering a file

sort re-orders lines in ASCII collating sequenceswhitespaces first, then numerals, uppercase and finally lowercase

we can sort the file based on a field by using -t and –k option.

$ sort -t" " -k 2 students.txt

/* sorts the file based on the second field

using the delimiter as space*/

grep : searching for a pattern

grep scans its input for a pattern, and can display the selected pattern, the line numbers or the filename where the pattern occurs.

usage: grep options pattern filename(s)

b)

NAME OF THE EXPERIMENT: Linux commands cp , mv using Linux system calls

AIM: Implement the Linux commands (a) cp (b) mv using Linux system calls.

ALGORITHM:

Step 1: Start

Step 2: open Terminal

Step 3:give the cp and mv command and check with the different type of the options

Step 4:observe the result of the each options

Step 5: Stop

Program:

cp :copy files and directories

Usage: cp [OPTION]... SOURCE DEST

eg. cp sample.txt sample_copy.txt

cp :sample_copy.txt target_dir

mv : move (rename) files

Usage: mv [OPTION]... SOURCE DEST

eg. mv source.txt target_dir

mv old.txt new.txt

VIVA-VOCE

1. **How would you kill a process?**

The **killall** command is used to kill processes by name.

2. **What are the different file types available ?**

In Linux, everything is considered as a file. In UNIX, seven standard file types are regular, directory, symbolic link, FIFO special, block special, character special, and socket.

3. **Explain the method of changing file access permission?**

To change the file or the directory permissions, you use the **chmod (change mode) command**. There are two ways to use chmod — the symbolic mode and the absolute mode.

4. Which are the Linux Directory Commands?

Directory Command	Description
pwd	The pwd command stands for (print working directory). It displays the current working location or directory of the user. It displays the whole working path starting with /. It is a built-in command.
ls	The ls command is used to show the list of a folder. It will list out all the files in the directed folder.
cd	The cd command stands for (change directory). It is used to change to the directory you want to work from the present directory.
mkdir	With mkdir command you can create your own directory.
rmdir	The rmdir command is used to remove a directory from your system.

5. How to rename a file in Linux using cp command?

If we want to rename and copy at the same time, then we use the following command.

cp program3.cpp homework6.cpp

6. How would you delete a directory in Linux?

1. To remove an empty directory, use either rmdir or rm -d followed by the directory name: rm -d dirname rmdir dirname.
2. To remove non-empty directories and all the files within them, use the rm command with the -r (recursive) option: rm -r dirname.

WEEK 9: NAME OF THE EXPERIMENT: SHELL SCRIPT

AIM:a) Write a shell script to find factorial of a given integer.

ALGORITHM:

Step 1: Start

Step 2: Read any number to find factorial

Step 3: initialize fact=1 and i=1

Step 4:while i less than do

fact=fact*i

i=i+1

Step 4:print fact

Step 5: stop

Source code :

```
echo enter a number
read a
i=2
fact=1
if [ $a -ge 2 ]
then
while [ $i -le $a ]
do
fact=`expr $fact \* $i`
i=`expr $i + 1`
done
fi
echo factorial of $a=$fact
```

output:

```
[latha@localhost ~]$ sh fact.sh
enter a number
5
factorial of 5=120
```

b) NAME OF THE EXPERIMENT:Creating a child process and allow the parent to display 'parent'.

AIM:Write a C program to create a child process and allow the parent to display 'parent' and

the child to display 'child' on the screen.

ALGORITHM:

Step 1: Start the main function

Step 2: call the fork() function to create a child process fork function returns 2 values

Step 3:which returns 0 to child process

Step 4:which returns process id to the parent process

Step 5:stop

Source Code:

```
include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

int global=10;

int main()

{

int local=20;

pid_t pid;

printf("before fork\n");

printf("pid=%d,global=%d,local=%d\n",getpid(),global,local);

pid=fork();

if(pid<0)

printf("failed to create child");

else if(pid==0)

{

printf("after fork\n");
```

```
global++;
local++;
}
else if(pid>0)
sleep(2);
printf("cid=%d,global=%d,local=%d\n",getpid(),global,local);
exit(0);
}
```

Output:

```
 [latha@localhost ~]$ cc week16.c
[latha@localhost ~]$ ./a.out
before fork
pid=3005,global=10,local=20
after fork
cid=3006,global=11,local=21
cid=3005,global=10,local=20
```

c)

NAME OF THE EXPERIMENT: Parent writes a message to a pipe and the child reads themessage.

AIM: Write a C program in which a parent writes a message to a pipe and the child reads themessage.

ALGORITHM:

Step 1: Start

Step 2: import libraryfiles

#include <errno.h>

#include<fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

Step 3:create an array of 2 size a[0] is for reading and a[1] is for writing over a pipe

Step 4:open pipe using pipe(a)

Step 5:write a string "Hi" in pipe using write() function

Step 6:read from pipe "Hi" message using read() function

Step 7:stop

SOURCE CODE:

```
#include <stdio.h>

#include<stdlib.h>

#include <sys/wait.h> /* contains prototype for wait */

int main(void)
```

```
{
int pid;
int status;
printf("Hello World!\n");
pid = fork( );

if(pid == -1) /* check for error in fork */
{
perror("bad fork");
exit(1);
}
if (pid == 0)
printf("I am the child process.\n");
else { wait(&status); /* parent waits for child to finish */
printf("I am the parent process.\n");
return 0;
}
}
```

OUTPUT:

```
student@202.sys14:~$ ./a.out
 Hello World!
I am the child process
I am the parent process.
student@202.sys14:~$
```

VIVA-VOCE

1. What is Shell Script?

A Shell Script is a text file that contains one or more commands.

2. What are the different type of variables used in a shell Script?

In Linux shell script we can use two types of variables :

o System defined variables

o User defined variables

3. Why are we using fork() function call?

System call fork() is used to create processes. It takes no arguments and returns a process ID. The purpose of fork() is to create a *new* process, which becomes the *child* process of the caller. After a new child process is created, *both* processes will execute the next instruction following the *fork()* system call.

4. Why do we create a child process?

Sometimes there is a need for a program to perform more than one function simultaneously. Since these jobs may be interrelated so two different programs to perform them cannot be created.

5. Why do we use piping?

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing.

6. what is IPC?

Inter-process communication (IPC) is a mechanism that allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them.

Week-10:

NAME OF THE EXPERIMENT:CPU Scheduling Techniques FCFS , Priority

AIM:Write C Programs to simulate the following CPU scheduling algorithms:

a)FCFS b) Priority

a) FCFS ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Set the waiting of the first process as '0' and its burst time as its turnaround time

Step 5: for each process in the Ready Q calculate

Waiting time for process(n)= waiting time of process(n-1)+Burst time of process(n-1)

Turn around time for Process(n)= waiting time of Process(n)+ Burst time forprocess(n)

Step 6: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

SOURCE CODE:

```
#include<stdio.h>

void main()

{

int pid[10],bt[10],wt[10],tat[10],n,twt=0,ttat=0,i;

float awt,atat;

printf("Enter no.of processes:");

scanf("%d",&n);

printf("\n Enter burst times:");

for(i=0;i<n;i++)

scanf("%d",&bt[i]);

wt[0]=0;

tat[0]=bt[0];

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

wt[i]=tat[i-1];

tat[i]=bt[i]+wt[i];

}

for(i=0;i<n;i++){

ttat= ttat+tat[i];

twt=twt+wt[i];
```

```
}
printf("\n PID \t BT \t WT \t TAT");
for(i=0;i<n;i++)
printf("\n %d\t%d\t%d\t%d",i+1,bt[i],wt[i],tat[i]);
awt=(float)twt/n;
atat=(float)ttat/n;
printf("\nAvg. Waiting Time=%f",awt);
printf("\nAvg. Turn around time=%f",atat);
}
```

Output:

Enter no.of processes:3

Enter burst times:4

13

25

Enter pid 1 2 3

PID	BT	WT	TAT
1	4	0	4
2	13	4	17
3	25	17	42

Avg.waiting time=7.00000

Avg turnaround time=21,00000

Process returned 32(0X2) execution time=20.112s

Pres any key to continue.

b) PRIORITY ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id, process priority and accept the CPU burst time

Step 4: Start the Ready Q according the highest priority by sorting according to highest to lowest priority.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

Waiting time for process(n)= waiting time of process (n-1)+Burst time of process(n-1)

Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

SOURCE CODE:

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

```
void main()
{
int pid[10],bt[10],pr[10],wt[10],tat[10],n,twt=0,ttat=0,i,j,t;
float awt,atat;
printf("Enter no.of processes:");
scanf("%d",&n);
printf("\n Enter burst times:");
for(i=0;i<n;i++)
scanf("%d",&bt[i]);
printf("\n Enter PID:");
for(i=0;i<n;i++)
scanf("%d",&pid[i]);
printf("\n Enter Priorities:");
for(i=0;i<n;i++)
scanf("%d",&pr[i]);
for(i=0;i<n;i++){
for(j=i+1;j<n;j++){
if(pr[i]>pr[j]){
t=pr[i];
pr[i]=pr[j];
```

```
pr[j]=t;
 t=bt[i];
bt[i]=bt[j];
bt[j]=t;
 t=pid[i];
pid[i]=pid[j];
pid[j]=t;
}}}
wt[0]=0;
tat[0]=bt[0];
for(i=1;i<n;i++){
wt[i]=tat[i-1];
tat[i]=bt[i]+wt[i];
}
for(i=0;i<n;i++){
ttat= ttat+tat[i];
twt=twt+wt[i];
}
printf("\n PID PRIORITY \t BT \t WT \t TAT");
for(i=0;i<n;i++)
```

```
printf("\n %d\t%d\t%d\t%d\t%d",pid[i],pr[i],bt[i],wt[i],tat[i]);

awt=(float)twt/n;

atat=(float)ttat/n;

printf("\nAvg. Waiting Time=%f",awt);

printf("\nAvg. Turn around time=%f",atat);

}
```

Output:

```
 Enter   noof Processes :4
Enter  Burst Times:2
6
4
5
Enter Priorities :3
2
6
1
PID      PRIORITY   BT    WT     TT
 5          1        5     0      5
 6          2        6     5      11
 3          3        2     11     13
 4          6        4      3     17

Average Waiting Time: 7.250000
AvG Turnaround Time: 11.500000
Process returned 32(0X20) execution time:86.108s
```

VIVA-VOCE

1.What is an Operating system?

Operating System (OS), program that manages a computer's resources, especially the allocation of those resources among other programs. Typical resources include the central processing unit (CPU), computer memory, file storage, input/output (I/O) devices, and network connections.

2.What is a process ?

A process is an ‘active’ entity, instead of a program, which is considered a ‘passive’ entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created)

3.What Are The Advantages of A Multiprocessor System?

The advantages of the multiprocessing system are: Increased Throughput − By increasing the number of processors, more work can be completed in a unit time. Cost Saving − Parallel system shares the memory, buses, peripherals etc. Multiprocessor system thus saves money as compared to multiple single systems.

4. Explain starvation and Aging

Starvation is the problem that occurs when high priority processes keep executing and low priority processes get blocked for indefinite time. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time. For example, if priority range from 127(low) to 0(high), we could increase the priority of a waiting process by 1 Every 15 minutes.

5.What are the functions of operating system?

An operating system has three main functions: (1) manage the computer's resources, such as the central processing unit, memory, disk drives, and printers, (2) establish a user interface, and (3) execute and provide services for applications software.

Week-11:

NAME OF THE EXPERIMENT:CPU Scheduling Techniques SJF, Round Robin

AIM:Write C Programs to simulate the following CPU scheduling algorithms:

a)SJF b) Round Robin

a) SJF ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

Waiting time for process(n)= waiting time of process (n-1)+Burst time of process(n-1)

Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 7: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

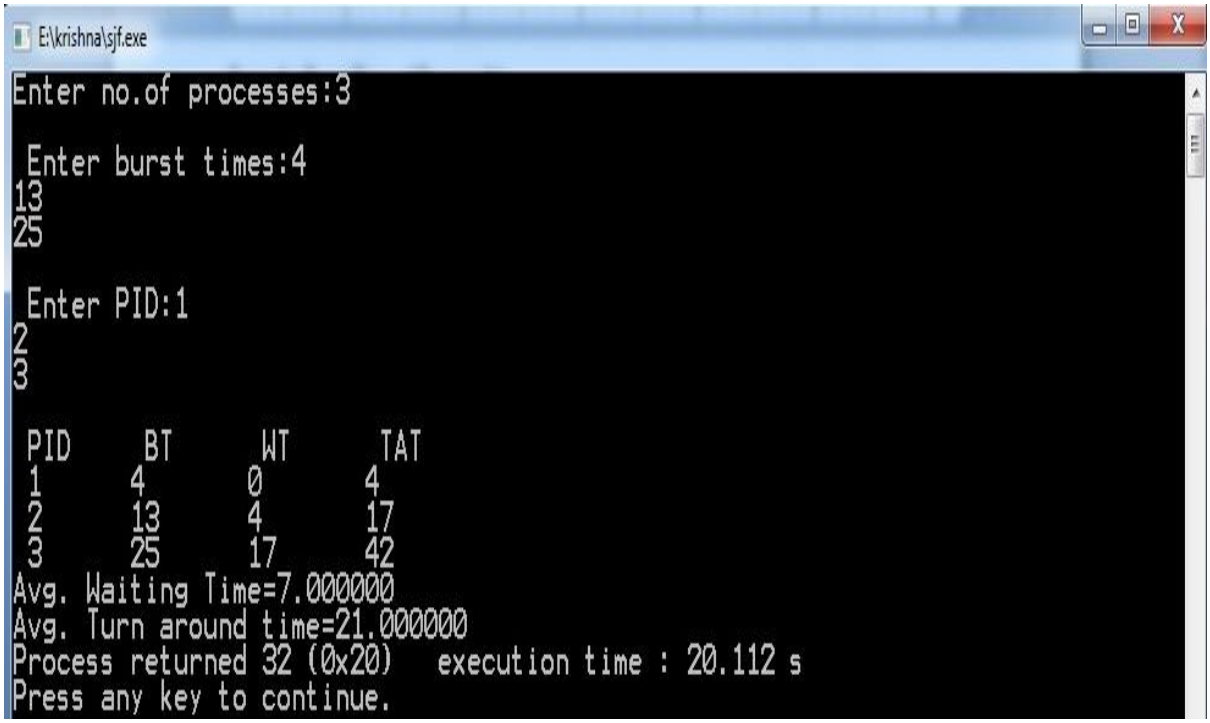
SOURCE CODE:

```
#include<stdio.h>

void main(){

int pid[10],bt[10],wt[10],tat[10],n,twt=0,ttat=0,i,j,t;

float awt,atat;

printf("Enter no.of processes:");

scanf("%d",&n);

printf("\n Enter burst times:");

for(i=0;i<n;i++)

scanf("%d",&bt[i]);

printf("\n Enter PID:");

for(i=0;i<n;i++)

scanf("%d",&pid[i]);

for(i=0;i<n;i++)

{

for(j=i+1;j<n;j++)

{

if(bt[i]>bt[j])

{

t=bt[i];
```

```
bt[i]=bt[j];
bt[j]=t;
 t=pid[i];
pid[i]=pid[j];
pid[j]=t;
}}}
wt[0]=0;
tat[0]=bt[0];
for(i=1;i<n;i++)
{
wt[i]=tat[i-1];
tat[i]=bt[i]+wt[i];
}
for(i=0;i<n;i++)
{
ttat= ttat+tat[i];
twt=twt+wt[i];
}
printf("\n PID \t BT \t WT \t TAT");
for(i=0;i<n;i++)
```

```
printf("\n %d\t%d\t%d\t%d",pid[i],bt[i],wt[i],tat[i]);

awt=(float)twt/n;

atat=(float)ttat/n;

printf("\nAvg. Waiting Time=%f",awt);

printf("\nAvg. Turn around time=%f",atat);

}
```

OUTPUT :



b) ROUND ROBIN ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue & time quantum or time slice

Step 3: For each process in the ready Q, assign the process id &accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where No. of time slice for process(n) = burst time process(n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

Waiting time for process(n) = waiting time of process(n-1)+burst time of process(n-1) + the time difference in getting the CPU from process(n-1)

Turn around time for process(n) = waiting time of process(n) + burst time of process(n) + the time difference in getting CPU from process(n).

Step 7: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

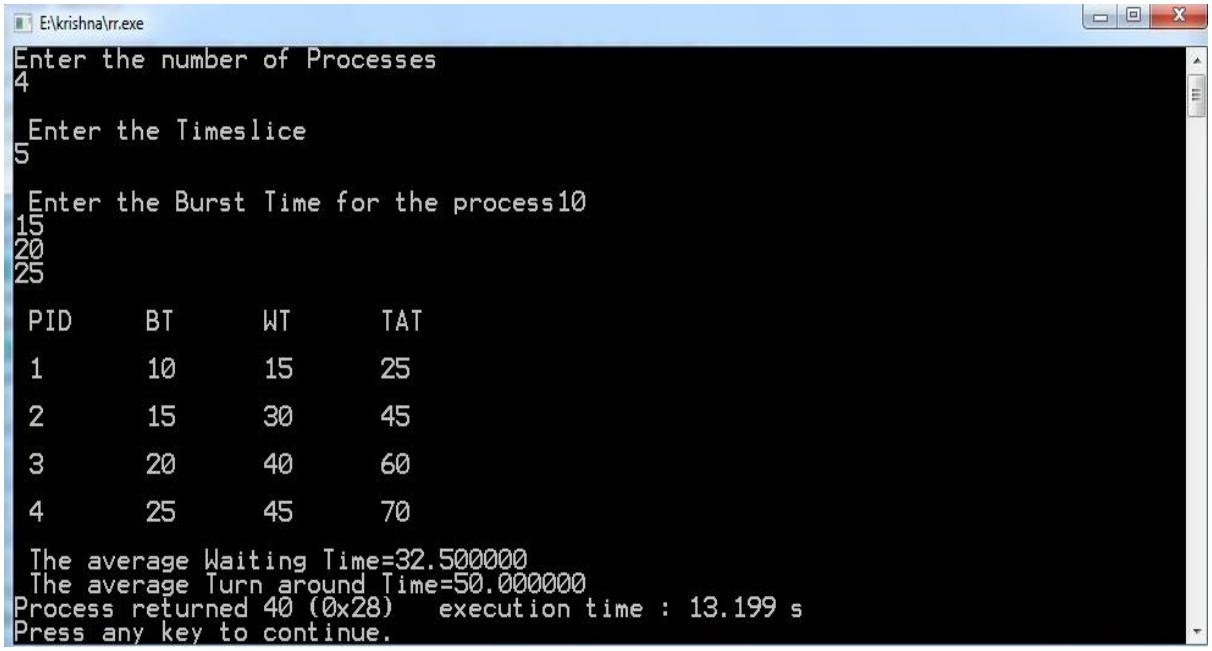
SOURCE CODE:

```
#include<stdio.h>

void main()

{

int ts,bt1[10],wt[10],tat[10],i,j=0,n,bt[10],ttat=0,twt=0,tot=0;

float awt,atat;

printf("Enter the number of Processes \n");

scanf("%d",&n);

printf("\n Enter the Timeslice \n");

scanf("%d",&ts);

printf("\n Enter the Burst Time for the process");

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

scanf("%d",&bt1[i]);

bt[i]=bt1[i];

}

while(j<n){

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

if(bt[i]>0){

if(bt[i]>=ts){
```

```
tot+=ts;
bt[i]=bt[i]-ts;
if(bt[i]==0){
j++;
tat[i]=tot;
}}
else{
tot+=bt[i];
bt[i]=0;
j++;
tat[i]=tot;
}}}}
for(i=1;i<=n;i++){
wt[i]=tat[i]-bt1[i];
twt=twt+wt[i];
ttat=ttat+tat[i];
}
awt=(float)twt/n;
atat=(float)ttat/n;
printf("\n PID \t BT \t WT \t TAT\n");
```

```
for(i=1;i<=n;i++) {

printf("\n %d \t %d \t %d \t %d \t\n",i,bt1[i],wt[i],tat[i]);

}

printf("\n The average Waiting Time=%f",awt);

printf("\n The average Turn around Time=%f",atat);

}
```

OUTPUT :



VIVA –VOCE

1. List different CPU Scheduling algorithms.

The different CPU algorithms are:

· First Come First Serve.

· Shortest Job First.

· Shortest Remaining Time First.

· Round Robin Scheduling.

· Priority Scheduling.

· Multilevel Queue Scheduling.

· Multilevel Feedback Queue Scheduling.

2. What is FCFS Scheduling?

First Come First Serve (FCFS) is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first.

3. What is SJF Scheduling?

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

4. What is RR Scheduling?

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU. One of the most commonly used technique in CPU scheduling as a core.

5. What is Priority Scheduling?

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis.

Week-12:

NAME OF THE EXPERIMENT: File allocation techniques

AIM: Write C programs to simulate the following Fileallocation strategies

a) Sequential b)Linked c)Indexed

a) Sequential Algorithm:

Step 1: Start the program.

Step 2: Get the number of memory partition and their sizes.

Step 3: Get the number of processes and values of block size for each process.

Step 4: First fit algorithm searches the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.

Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to requesting process and allocates if.

Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.

Step 7: Analyses all the three memory management techniques and display the best algorithm which utilizes the memory resources effectively and efficiently.

Step 8: Stop the program

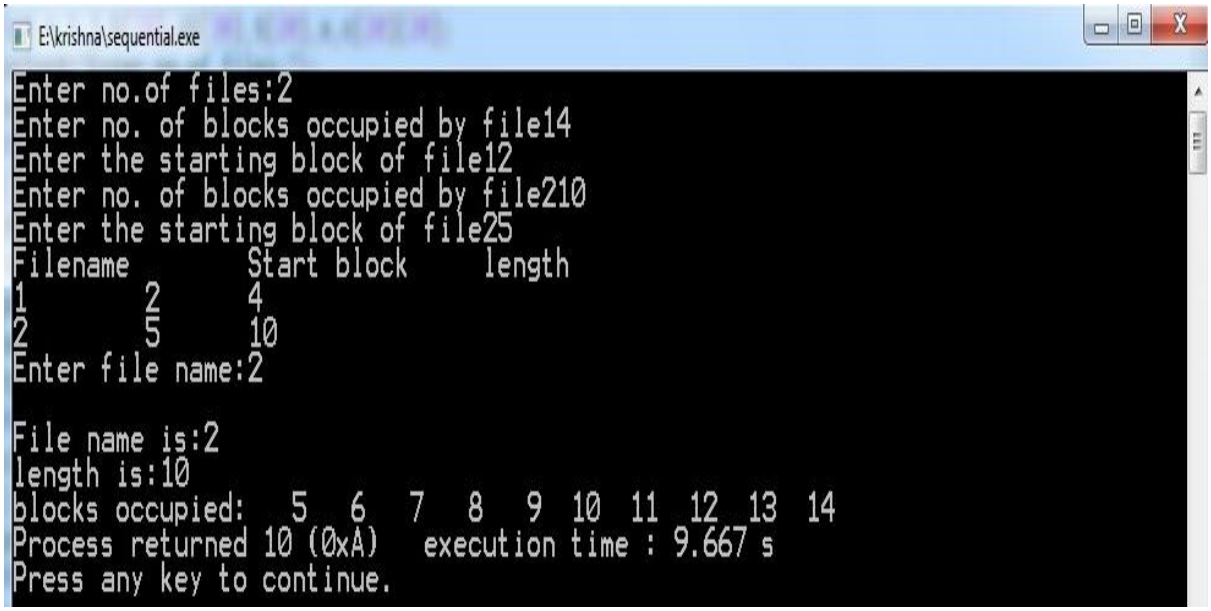
SOURCE CODE:

```
#include<stdio.h>

main()
```

```
{
int n,i,j,b[20],sb[20],t[20],x,c[20][20];
printf("Enter no.of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter no. of blocks occupied by file%d",i+1);
scanf("%d",&b[i]);
printf("Enter the starting block of file%d",i+1);
scanf("%d",&sb[i]);
t[i]=sb[i];
for(j=0;j<b[i];j++)
c[i][j]=sb[i]++;
}
printf("Filename\tStart block\tlength\n");
for(i=0;i<n;i++)
printf("%d\t %d \t%d\n",i+1,t[i],b[i]);
printf("Enter file name:");
scanf("%d",&x);
printf("\nFile name is:%d",x);
```

```
printf("\nlength is:%d",b[x-1]);

printf("\nblocks occupied:");

for(i=0;i<b[x-1];i++)

printf("%4d",c[x-1][i]);

}
```

Output:



b) LinkedAlgorithm:

Step 1: Start the program.

Step 2: Read the number of files

Step 3: For each file, read the file name, starting block and number of blocks and block numbers of the file.

Step 4: Start from the starting block and link each block of the file to the next block in a linked list fashion.

Step 5: Display the file name, starting block, size of the file & the blocks occupied by the file.

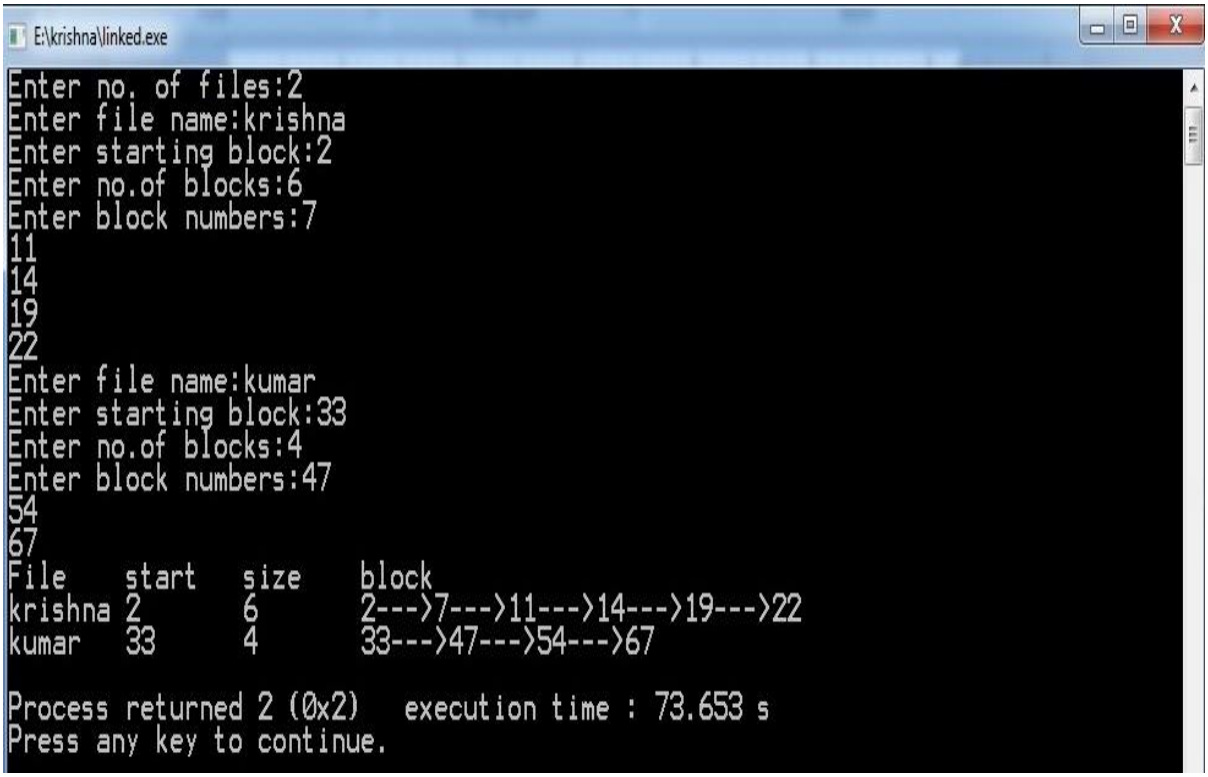
Step 6: Stop the program

SOURCE CODE:

```
#include<stdio.h>
struct file
{
char fname[10];
int start,size,block[10];
}f[10];
main()
{
int i,j,n;
printf("Enter no. of files:");
```

```
scanf("%d",&n);
for(i=0;i<n;i++){
printf("Enter file name:");
scanf("%s",&f[i].fname);
printf("Enter starting block:");
scanf("%d",&f[i].start);
f[i].block[0]=f[i].start;
printf("Enter no.of blocks:");
scanf("%d",&f[i].size);
printf("Enter block numbers:");
for(j=1;j<f[i].size;j++){
scanf("%d",&f[i].block[j]);
}}
printf("File\tstart\tsize\tblock\n");
for(i=0;i<n;i++){
printf("%s\t%d\t%d\t",f[i].fname,f[i].start,f[i].size);
for(j=0;j<f[i].size-1;j++)
printf("%d--->",f[i].block[j]);
printf("%d\n",f[i].block[j]);
}}
```

Output:


E:\krishna\linked.exe
Enter no. of files:2
Enter file name:krishna
Enter starting block:2
Enter no.of blocks:6
Enter block numbers:7
11
14
19
22
Enter file name:kumar
Enter starting block:33
Enter no.of blocks:4
Enter block numbers:47
54
67
File start size block
krishna 2 6 2--->7--->11--->14--->19--->22
kumar 33 4 33--->47--->54--->67
Process returned 2 (0x2) execution time : 73.653 s
Press any key to continue.

c) IndexedAlgorithm:

Step 1: Start the program.

Step2: Read the number of files

Step 3: Read the index block for each file.

Step 4: For each file, read the number of blocks occupied and number of blocks of the file.

Step 5: Link all the blocks of the file to the index block.

Step 6: Display the file name, index block, and the blocks occupied by the file.

Step 7: Stop the program

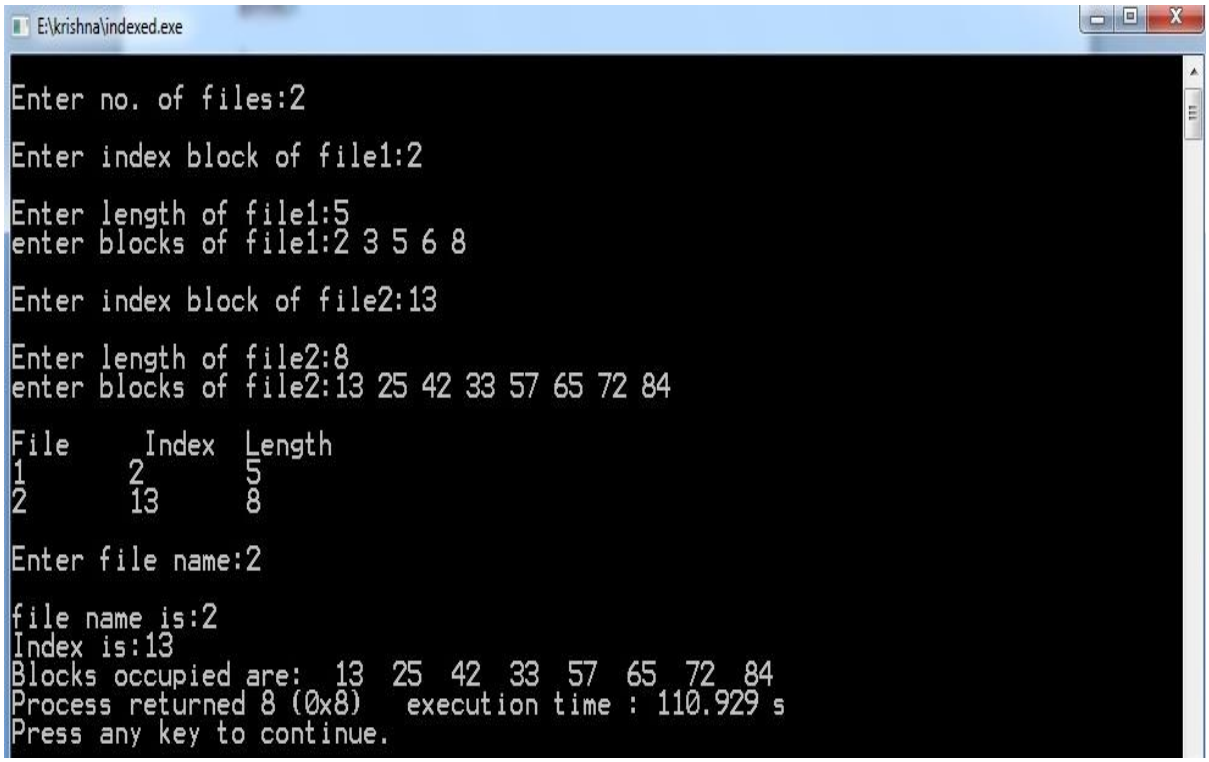
SOURCE CODE:

```
#include<stdio.h>
main()
{
int n,m[20],i,j,sb[20],b[20][20],x;
printf("\nEnter no. of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
```

```
printf("\nEnter index block of file%d:",i+1);
scanf("%d",&sb[i]);
printf("\nEnter length of file%d:",i+1);
scanf("%d",&m[i]);
printf("enter blocks of file%d:",i+1);
for(j=0;j<m[i];j++)
scanf("%d",&b[i][j]);
}
printf("\nFile\t Index\tLength\n");
for(i=0;i<n;i++)
{
printf("%d\t%d\t%d\n",i+1,sb[i],m[i]);
}
printf("\nEnter file name:");
scanf("%d",&x);
printf("\nfile name is:%d",x);
printf("\nIndex is:%d",sb[x-1]);
printf("\nBlocks occupied are:");
for(j=0;j<m[x-1];j++)
printf("%4d",b[x-1][j]);
```

}

OUTPUT:



VIVA-VOCE

1. List File access methods.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

· Sequential Access – It is the simplest access method. ...

· Direct Access – Another method is direct access method also known as relative access method. ...

· Index sequential method

2. Explain Sequential file allocation method.

In this allocation strategy, each file occupies a set of contiguously blocks on the disk. This strategy is best suited. For sequential files, the file allocation table consists of a single entry for each file. It shows the filenames, starting block of the file and size of the file.

3. Explain Linked file allocation method.

Each file is a linked list of disk blocks which need not be contiguous. The disk blocks canbe scattered anywhere on the disk.
The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

4. Explain Indexed file allocation method.

Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block. Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file. Directory entry will only contain the index block address.

5. What is file system mounting?

Mounting a file system attaches that file system to a directory (mount point) and makes it available to the system. The root (/) file system is always mounted. Any other file system can be connected or disconnected from the root (/) file system.

Week-13:

NAME OF THE EXPERIMENT:Memory management techniques

AIM: Write a C program to simulate the following memory management techniques

a) Paging b) Segmentation

a) PAGING Algorithm:

Step 1: Read all the necessary input from the keyboard.

Step 2: Pages - Logical memory is broken into fixed - sized blocks.

Step 3: Frames – Physical memory is broken into fixed – sized blocks.

Step 4: Calculate the physical address using the following

Physical address = (Frame number * Frame size) + offset

Step 5: Display the physical address.

Step 6: Stop the process

SOURCE CODE:

```
#include<stdio.h>
void main(){
int i,j,temp,framearr[20],pages,pageno,frames,memsize,log,pagesize,prosize,base;
printf("Enter the Process size: ");
scanf("%d",&prosize);
printf("\nEnter the main memory size: ");
scanf("%d",&memsize);
```

```
printf("\nEnter the page size: ");
scanf("%d",&pagesize);
pages=prosize/pagesize;
printf("\nThe process is divided into %d pages",pages);
frames = memsize/pagesize;
printf("\n\nThe main memory is divided into %d frames\n",frames);
for(i=0;i<frames;i++)
        framearr[i]=-1;         /* Initializing array elements with -1*/
for(i=0;i<pages;i++){
pos:    printf("\nEnter the frame number of page %d: ",i);
        scanf("%d",&temp);  /* storing frameno in temporary variable*/
        if(temp>=frames)  /*checking wether frameno is valid or not*/
        {
                printf("\n\t****Invalid frame number****\n");
                goto pos;
        }
        /* storing pageno (i.e 'i' ) in framearr at framno (i.e temp ) index */
        for(j=0;j<frames;j++)
                if(temp==j)
                        framearr[temp]=i;
```

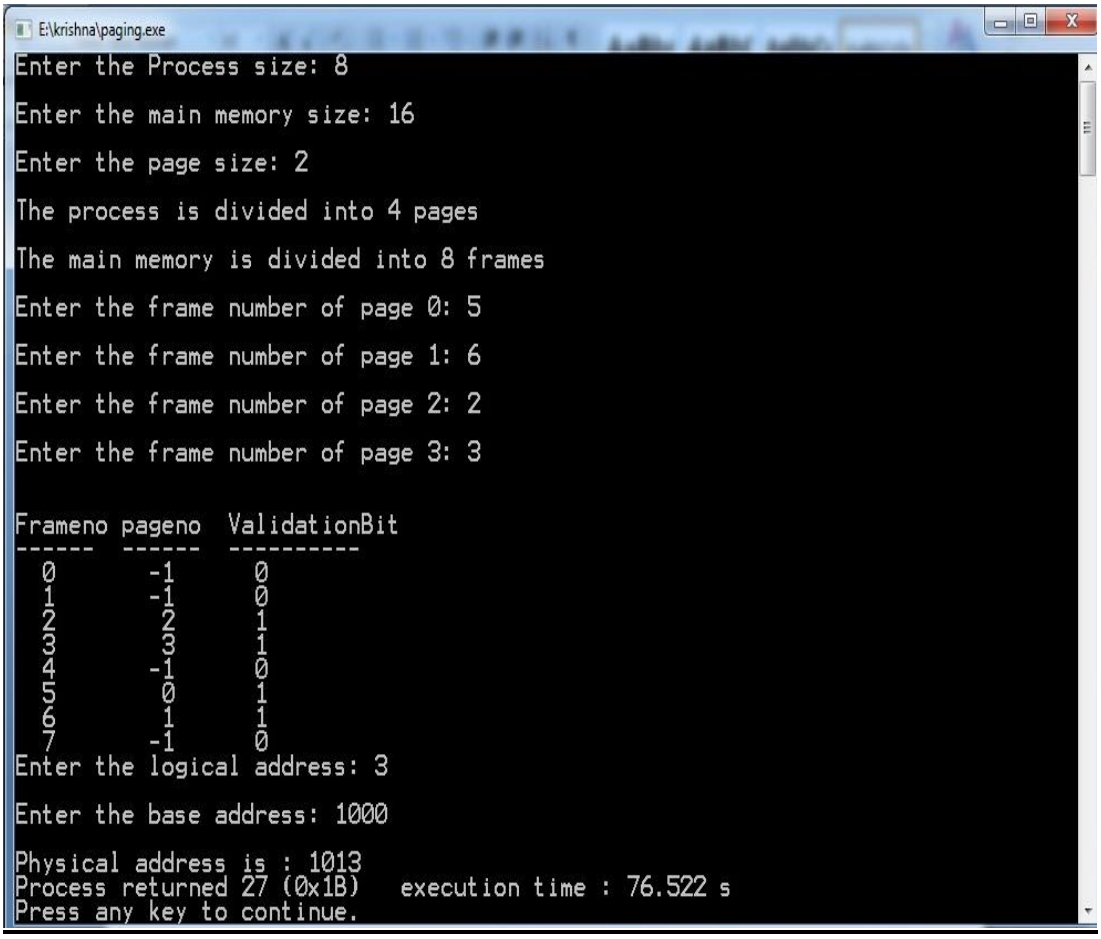
```
}
printf("\n\nFrameno\tpageno\tValidationBit\n------\t------\t----------");
for(i=0;i<frames;i++){
        printf("\n  %d  \t  %2d  \t",i,framearr[i]);
        if(framearr[i]==-1)
                printf("  0");
        else
                printf("  1");
}
printf("\nEnter the logical address: ");
scanf("%d",&log);
printf("\nEnter the base address: ");
scanf("%d",&base);
pageno = log/pagesize;
for(i=0;i<frames;i++)
        if(framearr[i]==pageno)
        {
                temp = i;
                break;
        }
```

```
        j = log%pagesize;       /* here 'j' is displacement */
```

temp = base + (temp*pagesize)+j; //lhs 'temp' is physical address rhs and 'temp' is frame num

```
        printf("\nPhysical address is : %d",temp);

        }
```

Output:



b) SEGMENTATION ALGORITHM

Step 1: Start the program.

Step 2: Get the number of segments.

Step 3: Get the base address and length for each segment.

Step 4: Get the logical address.

Step 5: Check whether the segment number is within the limit, if not display the error message.

Step 6: Check whether the byte reference is within the limit, if not display the error message.

Step 7: Calculate the physical memory and display it.

Step 8: Stop the program.

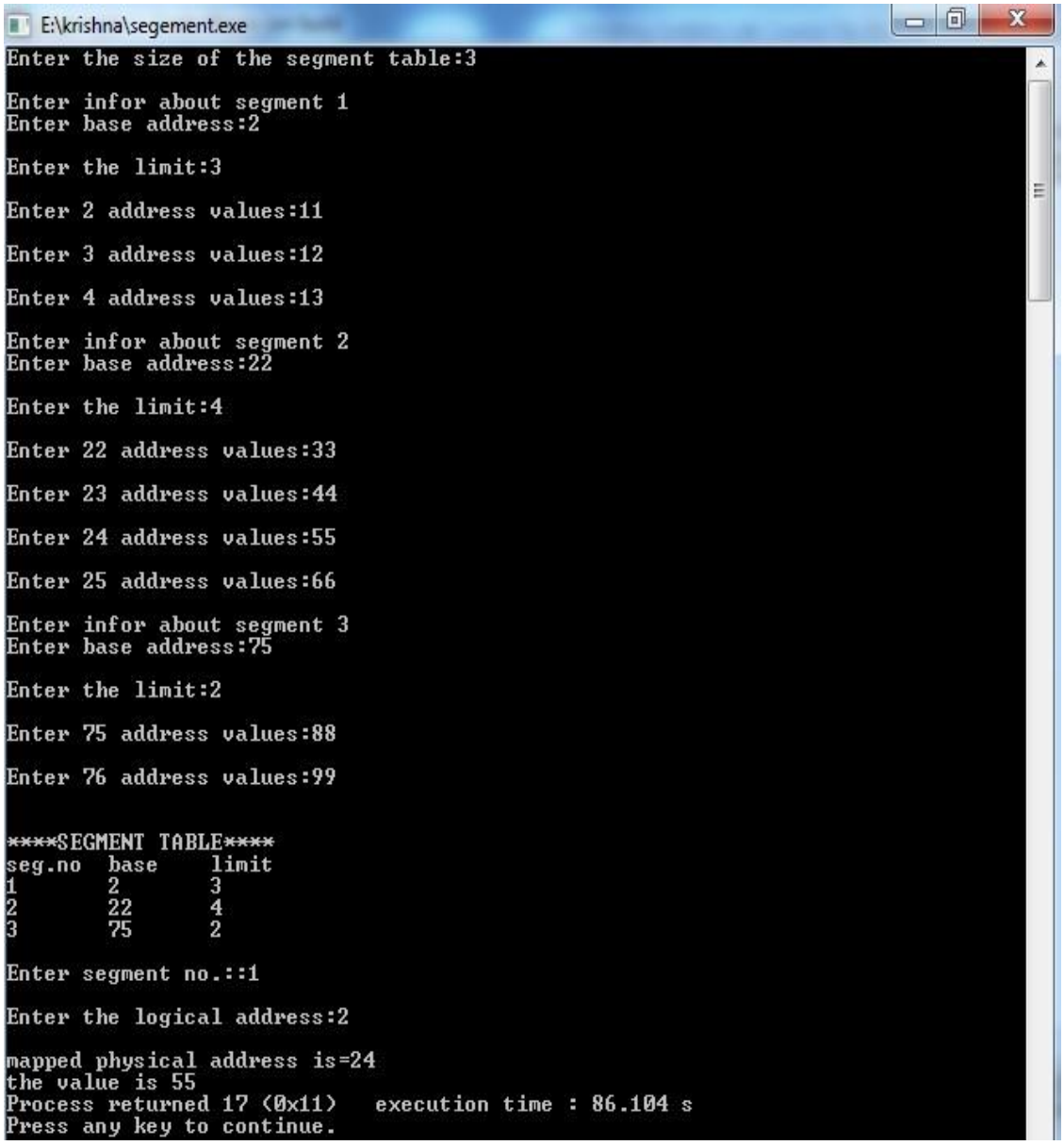
SOURCE CODE:

```
#include<stdio.h>
void main(){
    int i,j,m,size,val[10][10],badd[20],limit[20],ladd;
    printf("Enter the size of the segment table:");
    scanf("%d",&size);
    for(i=0;i<size;i++){
        printf("\nEnter infor about segment %d",i+1);
```

```
        printf("\nEnter base address:");
        scanf("%d",&badd[i]);
        printf("\nEnter the limit:");
        scanf("%d",&limit[i]);
        for(j=0;j<limit[i];j++){
                printf("\nEnter %d address values:",badd[i]+j);
                scanf("%d",&val[i][j]);
        }}
printf("\n\n****SEGMENT TABLE****");
printf("\nseg.no\tbase\tlimit\n");
for(i=0;i<size;i++)
{
        printf("%d\t%d\t%d\n",i+1,badd[i],limit[i]);
}
printf("\nEnter segment no.::");
scanf("%d",&i);
if(i>=size)
{
printf("invalid");
}
```

```
else
{
printf("\nEnter the logical address:");
scanf("%d",&ladd);
if(ladd>=limit[i])
printf("invalid");
else
{
m=badd[i]+ladd;
printf("\nmapped physical address is=%d",m);
printf("\nthe value is %d ",val[i][ladd]);
 }   }}
```

OUTPUT:


E:\krishna\segement.exe
Enter the size of the segment table:3
Enter infor about segment 1
Enter base address:2
Enter the limit:3
Enter 2 address values:11
Enter 3 address values:12
Enter 4 address values:13
Enter infor about segment 2
Enter base address:22
Enter the limit:4
Enter 22 address values:33
Enter 23 address values:44
Enter 24 address values:55
Enter 25 address values:66
Enter infor about segment 3
Enter base address:75
Enter the limit:2
Enter 75 address values:88
Enter 76 address values:99
****SEGMENT TABLE****
seg.no base limit
1 2 3
2 22 4
3 75 2
Enter segment no.::1
Enter the logical address:2
mapped physical address is=24
the value is 55
Process returned 17 (0x11) execution time : 86.104 s
Press any key to continue.

VIVA-VOCE

1.What is the basic function of paging?

Paging is a memory management scheme that permits the physical-address space of a process to be non contiguous. It avoids the considerable problem of having to fit varied sized memory chunks onto the backing store.

2.What is fragmentation?

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused.

3.What is thrashing?

Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault.

4.Differentiate between logical and physical address.

Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address.

5.Explain internal fragmentation and external fragmentation.

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes internal fragmentation**.**

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as external fragmentation

Week-14:

NAME OF THE EXPERIMENT:Page Replacement Techniques

AIM: Write C programs to simulate the following Page Replacement Techniques:

a) FIFO b) LRU c)OPTIMAL

a)FIFO Algorithm:

Step1: Start

Step2: Read the number of frames

Step3: Read the number of pages

Step4: Read the page numbers

Step5: Initialize the values in frames to -1

Step6:Allocate the pages in to frames in First in first out order.

Step7: Display the number of page faults.

Step8: Stop

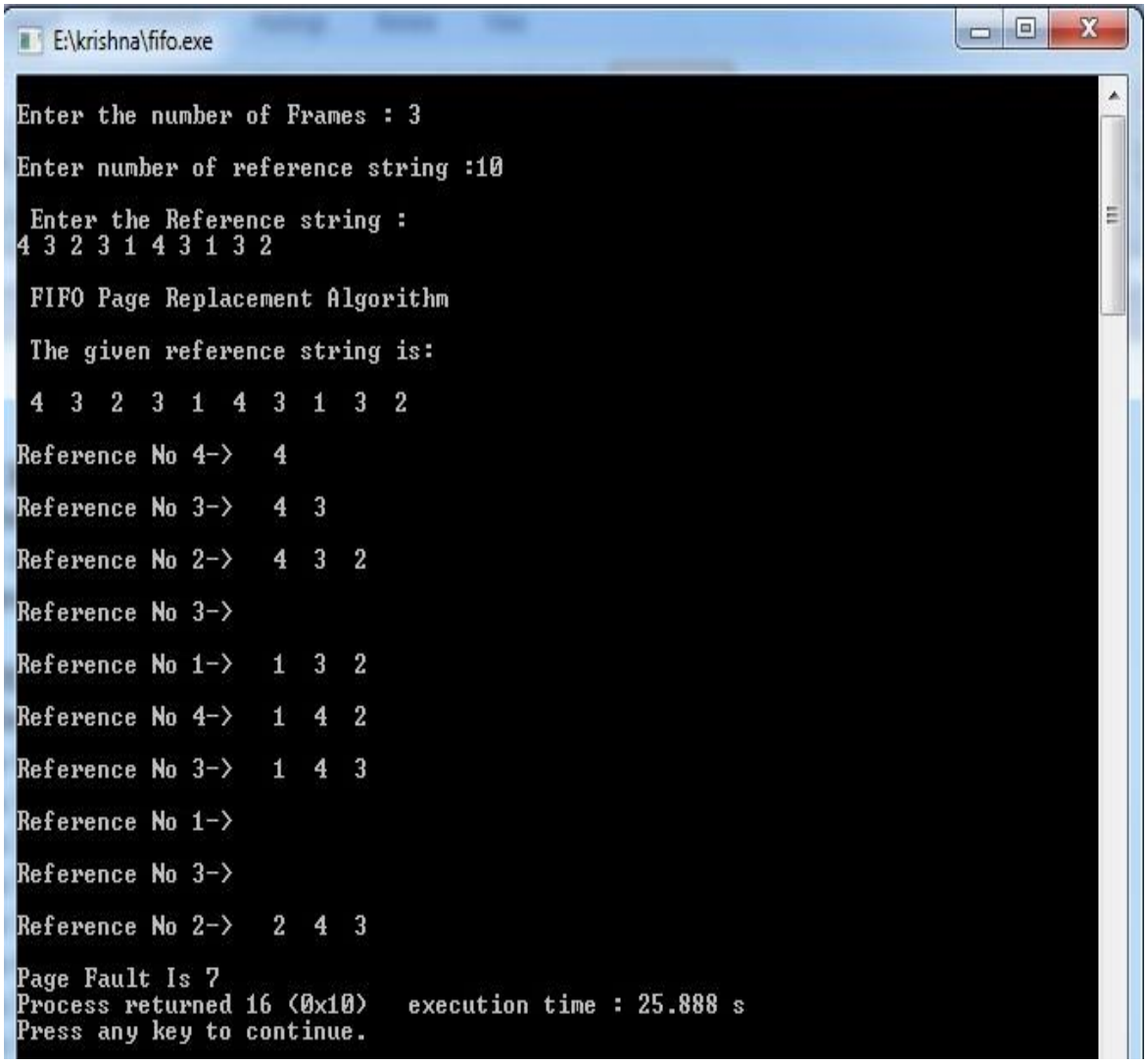
SOURCE CODE:

```
#include<stdio.h>
void main()
{
   int i,j,n,a[50],frame[10],fno,k,avail,pagefault=0;
printf("\nEnter the number of Frames : ");
scanf("%d",&fno);
```

```
printf("\nEnter number of reference string :");
scanf("%d",&n);
printf("\n Enter the Reference string :\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<fno;i++)
frame[i]= -1;
   j=0;
printf("\n FIFO Page Replacement Algorithm\n\n The given reference string is:\n\n");
for(i=0;i<n;i++)
   {
printf(" %d ",a[i]);
   }
printf("\n");
for(i=0;i<n;i++)
   {
printf("\nReference No %d-> ",a[i]);
avail=0;
for(k=0;k<fno;k++)
if(frame[k]==a[i])
```

```
avail=1;
if (avail==0)
        {
frame[j]=a[i];
            j = (j+1) % fno;
pagefault++;
for(k=0;k<fno;k++)
if(frame[k]!=-1)
printf(" %2d",frame[k]);
        }
printf("\n");
    }
printf("\nPage Fault Is %d",pagefault);
    }
```

Output:


E:\krishna\fifo.exe
Enter the number of Frames : 3
Enter number of reference string :10
Enter the Reference string :
4 3 2 3 1 4 3 1 3 2
FIFO Page Replacement Algorithm
The given reference string is:
4 3 2 3 1 4 3 1 3 2
Reference No 4-> 4
Reference No 3-> 4 3
Reference No 2-> 4 3 2
Reference No 3->
Reference No 1-> 1 3 2
Reference No 4-> 1 4 2
Reference No 3-> 1 4 3
Reference No 1->
Reference No 3->
Reference No 2-> 2 4 3
Page Fault Is 7
Process returned 16 (0x10) execution time : 25.888 s
Press any key to continue.

b) LRU Algorithm:

Step1: Start

Step2: Read the number of frames

Step3: Read the number of pages

Step4: Read the page numbers

Step5: Initialize the values in frames to -1

Step6: Allocate the pages in to frames by selecting the page that has not been used for the longest period of time.

Step7: Display the number of page faults.

Step8: Stop

SOURCE CODE:

```
#include<stdio.h>
void main()
{
   int i,j,l,max,n,a[50],frame[10],flag,fno,k,avail,pagefault=0,lru[10];
printf("\nEnter the number of Frames : ");
scanf("%d",&fno);
printf("\nEnter number of reference string :");
scanf("%d",&n);
```

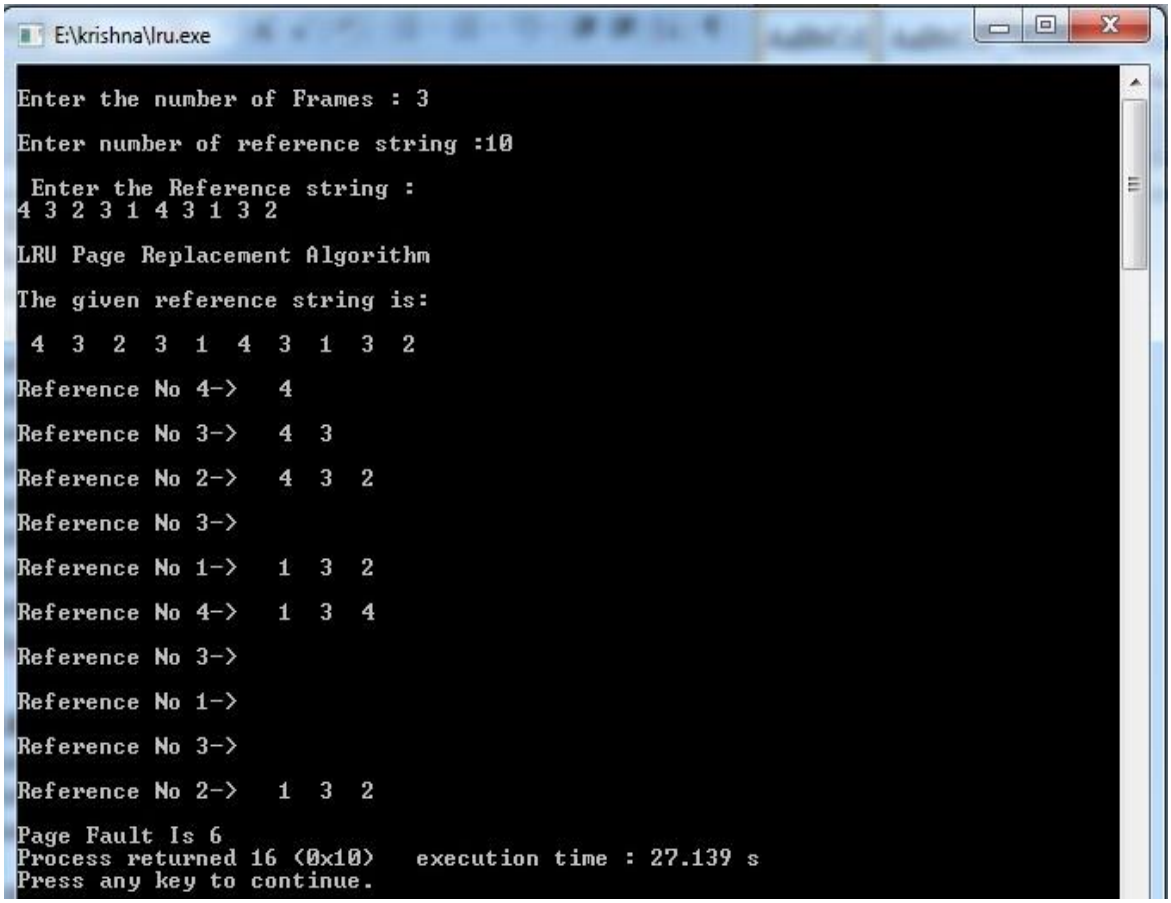
```
printf("\n Enter the Reference string :\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<fno;i++)
   {
frame[i]= -1;
lru[i] = 0;
   }
printf("\nLRU Page Replacement Algorithm\n\nThe given reference string is:\n\n");
for(i=0;i<n;i++)
   {
printf(" %d ",a[i]);
   }
printf("\n");
   j=0;
for(i=0;i<n;i++)
   {
max = 0;
flag=0;
printf("\nReference No %d-> ",a[i]);
```

```
avail=0;
for(k=0;k<fno;k++)
if(frame[k]==a[i])
        {
avail=1;
lru[k]=0;
break;
        }
if(avail==1)
    {
for(k=0;k<fno;k++)
if(frame[k]!=-1)
            ++lru[k];
max = 0;
for(k=1;k<fno;k++)
if(lru[k]>lru[max])
max = k;
        j = max;
    }
if(avail==0)
```

```
        {
lru[j]=0;
frame[j]=a[i];
for(k=0;k<fno;k++)
            {
if(frame[k]!=-1)
                    ++lru[k];
else
                {
                    j = k;
flag = 1;
break;
                }
            }
if(flag==0){
max = 0;
for(k=1;k<fno;k++)
if(lru[k]>lru[max])
max = k;
                j = max;
```

```
            }
pagefault++;
for(k=0;k<fno;k++)
if(frame[k]!=-1)
printf(" %2d",frame[k]);
        }
printf("\n");
    }
printf("\nPage Fault Is %d",pagefault);
}
```

OUTPUT :


E:\krishna\lru.exe
Enter the number of Frames : 3
Enter number of reference string :10
Enter the Reference string :
4 3 2 3 1 4 3 1 3 2
LRU Page Replacement Algorithm
The given reference string is:
4 3 2 3 1 4 3 1 3 2
Reference No 4-> 4
Reference No 3-> 4 3
Reference No 2-> 4 3 2
Reference No 3->
Reference No 1-> 1 3 2
Reference No 4-> 1 3 4
Reference No 3->
Reference No 1->
Reference No 3->
Reference No 2-> 1 3 2
Page Fault Is 6
Process returned 16 (0x10) execution time : 27.139 s
Press any key to continue.

c) Optimal Algorithm:

Step1: Start

Step2: Read the number of frames

Step3: Read the number of pages

Step4: Read the page numbers

Step5: Initialize the values in frames to -1

Step6: Allocate the pages in to frames by selecting the page that will not be used for the longest period of time.

Step7: Display the number of page faults.

Step8: Stop

SOURCE CODE :

```
#include<stdio.h>
int main()
{
   int i,j,l,min,flag1,n,a[50],temp,frame[10],flag,fno,k,avail,pagefault=0,opt[10];
printf("\nEnter the number of Frames : ");
scanf("%d",&fno);
printf("\nEnter number of reference string :");
```

```
scanf("%d",&n);
printf("\n Enter the Reference string :\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<fno;i++)
   {
frame[i]= -1;
opt[i]=0;
   }
printf("\nLFU Page Replacement Algorithm\n\nThe given reference string is:\n\n");
for(i=0;i<n;i++)
printf(" %d ",a[i]);
printf("\n");
   j=0;
for(i=0;i<n;i++)
   {
flag=0;
      flag1=0;
printf("\nReference No %d-> ",a[i]);
avail=0;
```

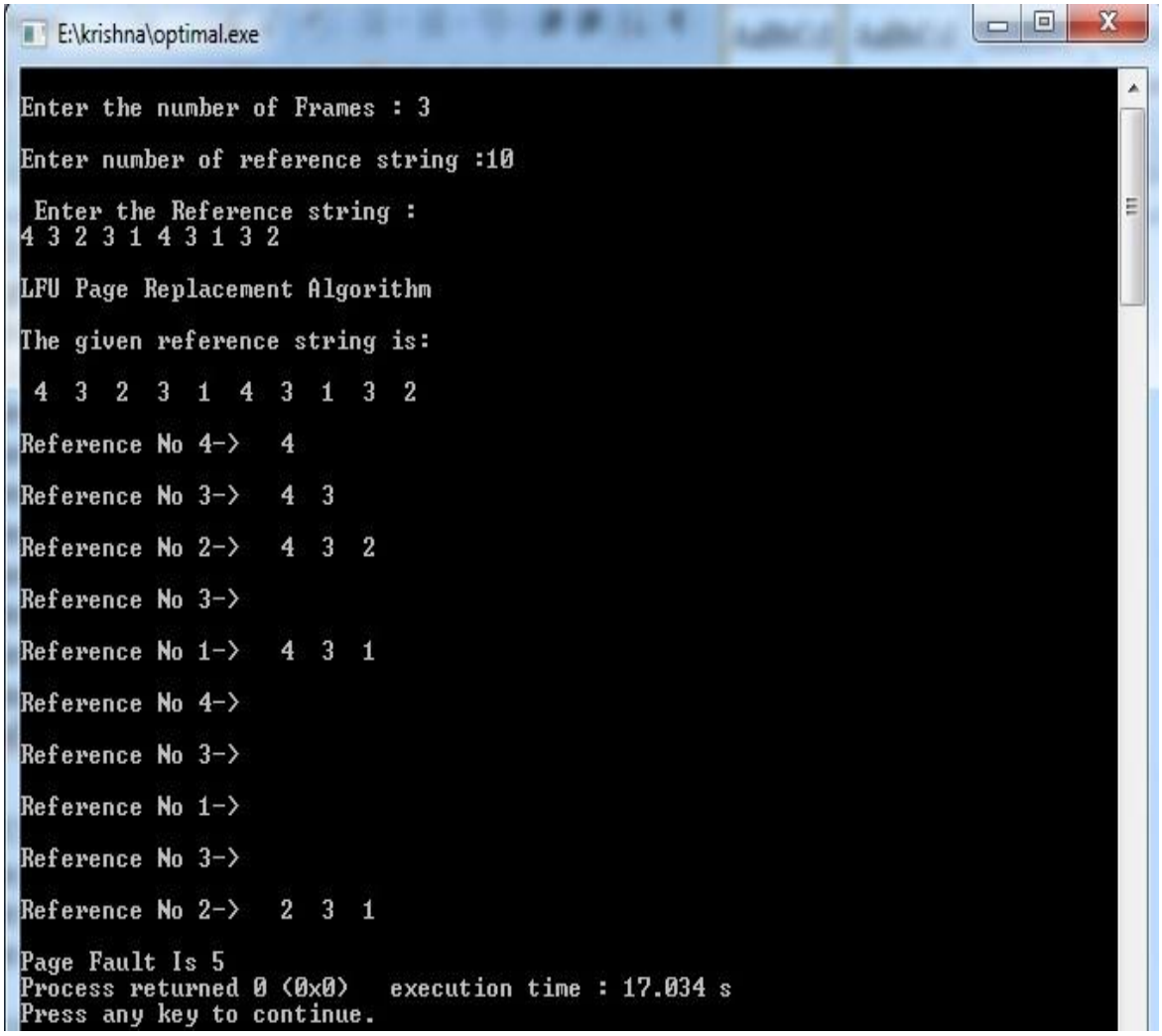
```
for(k=0;k<fno;k++)
if(frame[k]==a[i])
        {
avail=1;
break;
        }
if(avail==0)
    {
temp = frame[j];
frame[j]=a[i];
for(k=0;k<fno;k++)
        {
if(frame[k]==-1)
            {
               j = k;
flag = 1;
break;
            }
        }
if(flag==0)
```

```
        {
for(k=0;k<fno;k++)
            {
opt[k]=0;
for(l=i;l<n;l++)
                {
if(frame[k]==a[l])
                  {
                     flag1 = 1;
break;
                  }
                }
if(flag1==1)
opt[k] = l-i;
else
                {
opt[k] = -1;
break;
                }
            }
```

```
min = 0;
for(k=0;k<fno;k++)
if(opt[k]<opt[min]&&opt[k]!=-1)
min = k;
else if(opt[k]==-1)
                {
min = k;
frame[j] = temp;
frame[k] = a[i];
break;
                }
            j = min;
          }
pagefault++;
for(k=0;k<fno;k++)
if(frame[k]!=-1)
printf(" %2d",frame[k]);
       }
printf("\n");
   }
```

```
printf("\nPage Fault Is %d",pagefault);

return 0;

}
```

Output:



VIVA-VOCE

1.Why do we use page replacement algorithms?

Page replacement algorithms are an important part of virtual memory management and it helps the OS to decide which memory page can be moved out, making space for the currently needed page.However, the ultimate objective of all page replacement algorithms is to reduce the number of page faults

2.Which is best page replacement algorithm and why?

LRU resulted to be the best algorithm for page replacement to implement. LRU maintains a linked list of all pages in the memory, in which, the most recently used page is placed at the front, and the least recently used page is placed at the rear.

3.Explain LRU algorithm.

LRU stands for Least Recently Used.LRU replaces the line in the cache that has been in the cache the longest with no reference to it. It works on the idea that the more recently used blocks are more likely to be referenced again.

4. When does a page fault occur?

Page fault occurs when a requested page is mapped in virtual address space but not present in memory.

5.What are page replacement algorithms in OS?

1. First In First Out (FIFO)

2. Optimal Page replacement

3. Least Recently Used