Ex.No:3	**Process Management using System Calls** **(Programs using the following system calls of UNIX operating system fork, exec, getpid, exit, wait, close, opendir, readdir)**

1. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEMS (OPENDIR,READDIR, CLOSEDIR)

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<dirent.h>
struct dirent *dptr;
int main(int argc, char *argv[])
{
char buff[100];
DIR *dirp;
printf("\n\n ENTER DIRECTORY NAME");
scanf("%s", buff);
if((dirp=opendir(buff))==NULL)
{
printf("The given directory does not exist");
exit(1);
}
while(dptr=readdir(dirp))
{
printf("%s\n",dptr->d_name);
}
closedir(dirp);
}
```

SAMPLE OUTPUT

```
ENTER DIRECTORY NAME panimalar
.
    ..
```

2. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEM (fork, getpid, exit)

PROGRAM

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
void main()
{
int pid,pid1,pid2;
pid=fork();
if(pid==-1)
{
printf("ERROR IN PROCESS CREATION \n");
exit(1);
}
if(pid!=0)
{
pid1=getpid();
printf("\n the parent process ID is %d\n", pid1);
} else
{
pid2=getpid();
printf("\n the child process ID is %d\n", pid2);
} }
```

SAMPLE OUTPUT

```
 the parent process ID is 1512
 the child process ID is 1513
```

RESULT

Ex.No:4	SIMPLE SHELL PROGRAMS

1.Write a Shell program to check the given number is even or odd

PROGRAM

```
echo "Enter the Number"
read n
r=`expr $n % 2`
if [ $r -eq 0 ]
then
echo "$n is Even number"
else
echo "$n is Odd number"
fi
```

SAMPLE OUTPUT

```
Enter the Number4
4   is Even number
```

2.Write a Shell program to check the given year is leap year or not

PROGRAM

```
echo "Enter the year"
read y
b=`expr $y % 4`
if [ $b -eq 0 ]
then
echo "$y is a leap year"
else
echo "$y is not a leap year"
fi
```

SAMPLE OUTPUT

```
Enter the year
2012
2012 is a leap year
```

3. Write a Shell program to find the factorial of a number

PROGRAM

```
echo "Enter a Number"
read n
i=`expr $n - 1`
p=1
while [ $i -ge 1 ]
do
n=`expr $n \* $i`
i=`expr $i - 1`
done
echo "The Factorial of the given Number is $n"
```

SAMPLE OUTPUT

```
 Enter a Number
 4
The Factorial of the given Number is 24
```

4. Write a Shell program to swap the two integers

PROGRAM

```
    echo "Enter Two Numbers"
    read a b
    temp=$a
    a=$b
    b=$temp
    echo "after swapping"
    echo $a $b
```

SAMPLE OUTPUT

```
   Enter Two Numbers

   4 3
   after swapping
   3 4
```

Result:

Ex.No:5.A	**CPU SCHEDULING ALGORITHMS**
	FCFS

PROGRAM

```
    #include<stdio.h>
    #include<stdlib.h>
    struct fcfs
    {
    int  pid;
    int btime;
    int wtime;
    int ttime;
    }
    p[10];
    int main()
    {
    int i,n;
    int totwtime=0,totttime=0;
    printf("\n fcfs scheduling...\n");
    printf("enter the no of process");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
    p[i].pid=1;
    printf("\n burst time of the process");
    scanf("%d",&p[i].btime);
    }
    p[0].wtime=0;
    p[0].ttime=p[0].btime;
    totttime+=p[i].ttime;
    for(i=0;i<n;i++){
  p[i].wtime=p[i-1].wtime+p[i-1].btime;
  p[i].ttime=p[i].wtime+p[i].btime;
  totttime+=p[i].ttime;
  totwtime+=p[i].wtime;
}
 for(i=0;i<n;i++)
{{
printf("\n waiting time for process");
printf("\n turn around time for process");
printf("\n");
}}
printf("\n total waiting time :%d", totwtime );
printf("\n average waiting time :%f",(float)totwtime/n);
printf("\n total turn around time :%d",totttime);
printf("\n average turn around time: :%f",(float)totttime/n);
}
```

SAMPLE OUTPUT

```
        fcfs scheduling...
        enter the no of process 2

         burst time of the process 1

         burst time of the process 4


         waiting time for process turn
         around time for process

         waiting time for process
         turn around time for process

         total waiting time :1
         average waiting time :0.500000
         total turn around time :6

         average turn around time: :3.000000
```

RESULT

Ex.No:5.B	**CPU SCHEDULING ALGORITHMS**
	SJF SCHEDULING

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
typedef struct
{
int  pid;
int btime;
int wtime;
}
sp;
int main()
{
int i,j,n,tbm=0,totwtime=0,totttime;
sp *p,t;
printf("\n sjf schaduling ..\n");
printf("enter the no of processor");
scanf("%d",&n);
p=(sp*)malloc(sizeof(sp));
printf("\n enter the burst time");
for(i=0;i<n;i++)
{
printf("\n process %d\t",i+1);
scanf("%d",&p[i].btime);
p[i].pid=i+1;
p[i].wtime=0;
}
for(i=0;i<n;i++)
for(j=j+1;j<n;j++)
{
if(p[i].btime>p[j].btime)
{
t=p[i];
p[i]=p[j];
p[j]=t;
}}
printf("\n process scheduling\n");
printf("\n process \tburst time \t waiting time");
for(i=0;i<n;i++)
{
```

```
totwtime+=p[i].wtime=tbm;
tbm+=p[i].btime;
printf("\n%d\t\t%d",p[i].pid,p[i].btime);
printf("\t\t%d\t\t%d",p[i].wtime,p[i]);
}
totttime=tbm+totwtime;
printf("\n total waiting time :%d", totwtime );
printf("\n average waiting time :%f",(float)totwtime/n);
printf("\n total turn around time :%d",totttime);
printf("\n average turn around time: :%f",(float)totttime/n);
}
```

SAMPLE OUTPUT

```
                  sjf schaduling ..
 enter the no of processor 2


  enter the burst time
  process 1      4
  process 2      2
  process scheduling

  process        burst time       waiting time
 1             4              0              1
 2             2              4              2

  total waiting time :4
  average waiting time :2.000000
  total turn around time :10

  average turn around time: :5.000000
```

RESULT

Ex.No:5C	CPU SCHEDULING ALGORITHMS
	PRIORITY

PROGRAM

```
#include<stdio.h>
#include<stdio.h>
#include<stdlib.h>
typedef struct
{
int pno;
int  pri;
int btime;
int wtime;
}sp;
int main()
{
int i,j,n;
int tbm=0,totwtime=0,totttime=0; sp *p,t;
printf("\n PRIORITY SCHEDULING.\n");
printf("\n enter the no of process.       \n");
scanf("%d",&n); p=(sp*)malloc(sizeof(sp));
printf("enter the burst time and priority:\n"); for(i=0;i<n;i++)
{
printf("process%d:",i+1);
scanf("%d%d",&p[i].btime,&p[i].pri);
p[i].pno=i+1;
p[i].wtime=0;
}
for(i=0;i<n-1;i++) for(j=i+1;j<n;j++)
{
if(p[i].pri>p[j].pri)
{
t=p[i]; p[i]=p[j]; p[j]=t;
}
}
printf("\n process\tbursttime\twaiting time\tturnaround time\n");
for(i=0;i<n;i++)
{
totwtime+=p[i].wtime=tbm; tbm+=p[i].btime;
printf("\n%d\t\t%d",p[i].pno,p[i].btime);
printf("\t\t%d\t\t%d",p[i].wtime,p[i].wtime+p[i].btime);
}
totttime=tbm+totwtime;
printf("\n total waiting time:%d",totwtime);
printf("\n average waiting time:%f",(float)totwtime/n);
printf("\n total turnaround time:%d",totttime);
printf("\n avg turnaround time:%f",(float)totttime/n);
}
```

SAMPLE OUTPUT

```
PRIORITY SCHEDULING.

enter the no of process.
2
enter the burst time and priority:
process1:1
3
process2:5
5

process       bursttime       waiting time    turnaround time

1             1             0             1
2             5             1             6
total waiting time:1
average waiting time:0.500000
total turnaround time:7
avg turnaround time:3.500000
```

RESULT

Ex.No:5.D	CPU SCHEDULING ALGORITHMS
	ROUND ROBIN SCHEDULING

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
struct rr
{
int pno,btime,sbtime,wtime,lst;
}p[10];
int main()
{
int pp=-1,ts,flag,count,ptm=0,i,n,twt=0,totttime=0;
printf("\n round robin scheduling...........");
printf("enter no of processes:");
scanf("%d",&n);
printf("enter the time slice:");
scanf("%d",&ts);
printf("enter the burst time");
for(i=0;i<n;i++)
{
printf("\n process%d\t",i+1);
scanf("%d",&p[i].btime);
p[i].wtime=p[i].lst=0;
p[i].pno=i+1;
p[i].sbtime=p[i].btime;
}

printf("scheduling... \n");

do
{
flag=0;
for(i=0;i<n;i++)
{
count=p[i].btime;
if(count>0)
{
flag=-1;
count=(count>=ts)?ts:count;
printf("\n process %d",p[i].pno);
printf("from%d",ptm);
ptm+=count;
printf("to%d",ptm);
p[i].btime-=count;
if(pp!=i)
{
pp=i;
p[i].wtime+=ptm-p[i].lst-count;
p[i].lst=ptm;
}
}}
}
}
```

SAMPLE OUTPUT

```
ROUND ROBIN SCHEDULING.

enter the no of process.
2
enter the burst time and priority:
process1:1
3
process2:5
5

process       bursttime      waiting time    turnaround time

1             1              0               1
2             5              1               6
total waiting time:1
average waiting time:0.500000
total turnaround time:7
avg turnaround time:3.500000
```

RESULT

Ex.No:6	**IPC USING SHARED MEMORY**

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void  *shared_memory;
char buff[100], *segptr;
int shmid;
shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);
printf("Enter some data to write to shared memory\n");
read(0,buff,100);
strcpy(shared_memory,buff);
printf("You wrote : %s\n",(char *)shared_memory);
printf("Writing data to shared memory…\n");
strcpy(segptr,buff);
printf("DONE\n");
printf("Reading data from shared memory…\n");
printf("DATA:-%s\n",segptr);  printf("DONE\n");
printf("Removing shared memory Segment…\n");
if(shmctl(shmid,IPC_RMID,0)== -1)
printf("Can‟t Remove Shared memory Segment…\n");
else
printf("Removed Successfully");
}
```

SAMPLE OUTPUT

```
    Key of shared memory is -1
    Process attached at 0xffffffff
    Enter some data to write to shared memory
    panimalar
```

RESULT

Ex.No:7	**PRODUCER CONSUMER PROBLEM** **(MUTUAL EXCLUSION USING SEMAPHORES)**

PROGRAM

```
    #include<stdio.h>
    #include <stdlib.h>
    int mutex=1,full=0,empty=3,x=0;
    void main()
    {
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n1.PRODUCER\n2.CONSUMER\n3.EXIT\n");
    while(1) {
    printf("\nENTER YOUR CHOICE\n");
    scanf("%d",&n);
    switch(n)
    { case 1:
    if((mutex==1)&&(empty!=0))
    producer();
    else
    printf("BUFFER IS FULL");
    break;
    case 2:
    if((mutex==1)&&(full!=0))
    consumer();
printf("BUFFER IS EMPTY");
    break;
    case 3:
    exit(0);
    break;
    } } }
    int wait(int s) {
    return(--s); }
    int signal(int s) {
    return(++s); }
    void producer() {
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
    printf("\nproducer produces the item%d",x);
    mutex=signal(mutex); }
    void consumer() {
    mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\n consumer consumes item%d",x);
x--;
mutex=signal(mutex); }
```

SAMPLE OUTPUT

```
1.PRODUCER
2.CONSUMER
3.EXIT
ENTER YOUR CHOICE
1
producer produces the item1
ENTER YOUR CHOICE
1
producer produces the item2
ENTER YOUR CHOICE
2
 consumer consumes item2
ENTER YOUR CHOICE
3
```

RESULT

Ex.No:8	BANKERS ALGORITHM FOR DEADLOCK AVOIDANCE

PROGRAM

```
  #include<stdio.h>
  #include<stdlib.h>
  int  max[100][100];
  int alloc[100][100];
  int need[100][100];
  int avail[100];
  int n,r;
  void input();
  void show();
  void cal();
  int main()
  {
  int i,j;
  printf("********** Banker's Algo ************\n");
  input();
  show();
  cal();

  return 0;
  }
  void input()
  {
  int i,j;
  printf("Enter the no of Processes\t");
  scanf("%d",&n);
printf("Enter the no of resources instances\t");
scanf("%d",&r);
  printf("Enter the Max Matrix\n");
  for(i=0;i<n;i++)
  {
  for(j=0;j<r;j++)
  {
  scanf("%d",&max[i][j]);
  }}
  printf("Enter the Allocation Matrix\n");
  for(i=0;i<n;i++)
  {
  for(j=0;j<r;j++)
  {
  scanf("%d",&alloc[i][j]);
  }}
  printf("Enter the available Resources\n");
  for(j=0;j<r;j++)
  {
  scanf("%d",&avail[j]);
  }}
  void show()
  {
  int i,j;
  printf("Process\t Allocation\t Max\t Available\t");
  for(i=0;i<n;i++)
  {
  printf("\nP%d\t ",i+1);
  for(j=0;j<r;j++)
  {
  printf("%d ",alloc[i][j]);
  }
  printf("\t");
  for(j=0;j<r;j++)
  {
  printf("%d ",max[i][j]);
  }
  printf("\t");
  if(i==0)
  {
  for(j=0;j<r;j++)
  printf("%d ",avail[j]);
  }}}
  void cal()
  {
  int finish[100],temp,need[100][100],flag=1,k,c1=0;
int safe[100];
 int i,j;
for(i=0;i<n;i++)
  {
  finish[i]=0;
  }
  //find need matrix
  for(i=0;i<n;i++)
  {
  for(j=0;j<r;j++)
  {
  need[i][j]=max[i][j]-alloc[i][j];
  }}
  printf("\n");
  while(flag)
```

```
{
flag=0;
for(i=0;i<n;i++)
{
int c=0;
for(j=0;j<r;j++)
{
if((finish[i]==0)&&(need[i][j]<=avail[j]))
{
c++;
if(c==r)
{
for(k=0;k<r;k++)
{
avail[k]+=alloc[i][j];
finish[i]=1;
flag=1;
}
printf("P%d->",i);
if(finish[i]==1)
{
i=n;
}}}}}}
for(i=0;i<n;i++)
{
if(finish[i]==1)
{
c1++;
}
else
{printf("P%d->",i);}}
if(c1==n)
{printf("\n The system is in safe state");
}
else
{
printf("\n Process are in dead lock");
printf("\n System is in unsafe state");
}}
```

SAMPLE OUTPUT

```
********** Banker's Algo ************
Enter the no of Processes       2
Enter the no of resources instances     1
Enter the Max Matrix
2 2
Enter the Allocation Matrix
1 5
Enter the available Resources
4 5
Process  Allocation      Max     Available
P1       1      2       4
P2       5      2
P0->P1->
 The system is in safe state
```

RESULT

Ex.No:9	ALGORITHM FOR DEADLOCK DETECTION

PROGRAM

```
  #include<stdio.h>
  int max[100][100];
  int alloc[100][100];
  int need[100][100];
  int avail[100];
  int n,r;
  void input();
  void show();
  void cal();
  int main()
  {
  int i,j;
  printf("********** Deadlock Detection Algo ************\n");
  input();
  show();
  cal();

  return 0;
  }
  void input()
  {int i,j;
  printf("Enter the no of Processes\t");
  scanf("%d",&n);
  printf("Enter the no of resource instances\t");
  scanf("%d",&r);
  printf("Enter the Max Matrix\n");
for(i=0;i<n;i++)
  {for(j=0;j<r;j++)
  {
  scanf("%d",&max[i][j]);
  }}
  printf("Enter the Allocation Matrix\n");
  for(i=0;i<n;i++)
  {for(j=0;j<r;j++)
  {
  scanf("%d",&alloc[i][j]);
  }}
  printf("Enter the available Resources\n");
  for(j=0;j<r;j++)
  {
  scanf("%d",&avail[j]);
```

```
}}
void show()
{
int i,j;
printf("Process\t Allocation\t Max\t Available\t");
for(i=0;i<n;i++)
{
printf("\nP%d\t ",i+1);
for(j=0;j<r;j++)
{
printf("%d ",alloc[i][j]);
}
printf("\t");
for(j=0;j<r;j++)
{printf("%d ",max[i][j]);
}
printf("\t");
if(i==0)
{
for(j=0;j<r;j++)
printf("%d ",avail[j]);
}}}
void cal()
{ int finish[100],temp,need[100][100],flag=1,k,c1=0;
int dead[100];
int safe[100];
int i,j;
for(i=0;i<n;i++)
{finish[i]=0;
}
//find need matrix
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
need[i][j]=max[i][j]-alloc[i][j];
}}
while(flag)
{flag=0;
for(i=0;i<n;i++)
{int c=0;
for(j=0;j<r;j++)
{if((finish[i]==0)&&(need[i][j]<=avail[j]))
{c++;
if(c==r)
{
for(k=0;k<r;k++)
{avail[k]+=alloc[i][j];
finish[i]=1;
flag=1;
}//printf("\nP%d",i);
if(finish[i]==1)
{i=n;
}}}}}}
j=0;
flag=0;
for(i=0;i<n;i++)
{
if(finish[i]==0)
{dead[j]=i;
j++;
flag=1;
}}
if(flag==1)
{
printf("\n\nSystem is in Deadlock and the Deadlock process are\n");
for(i=0;i<n;i++)
{printf("P%d\t",dead[i]);
}}
else
{
printf("\nNo Deadlock Occur");
}
}
```

SAMPLE OUTPUT

```
********** Deadlock Detection Algo ************

Enter the no of Processes          1
        Enter the no of resource
        instances                          1
        Enter the Max Matrix
1
        Enter the Allocation
        Matrix2

Enter the available Resources4
        Process  Allocation        Max
         AvailableP1      2         1       4

        No Deadlock Occur
```

RESULT

Ex.No:10	THREADING & SYNCHRONIZATION APPLICATIONS

PROGRAM

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_t tid[2];
void* doSomeThing(void *arg)
{
    unsigned long i = 0;
    pthread_t id = pthread_self();

    if(pthread_equal(id,tid[0]))
    {
        printf("\n First thread processing\n");
    }
    else
    {
        printf("\n Second thread processing\n");
    }


    for(i=0; i<(0xFFFFFFFF);i++);
    return NULL;
   }
   int main(void)
   {
    int i = 0;
    int err;
    while(i < 2)
    {
        err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL);
        if (err != 0)
            printf("\ncan't create thread :[%s]", strerror(err));
        else
            printf("\n Thread created successfully\n");
        i++;
    }
    sleep(5);
    return 0;
}
```

SAMPLE OUTPUT

```
Thread created successfully

First thread processing

Thread created successfully

Second thread processing
```

RESULT

Ex.No:11	PAGING TECHNIQUE OF MEMORY MANAGEMENT

PROGRAM

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_t tid[2];
int counter;
pthread_mutex_t lock;
void* doSomeThing(void *arg)
{
pthread_mutex_lock(&lock);
 unsigned long i = 0;
counter += 1;
printf("\n Job %d started\n", counter);
for(i=0; i<(0xFFFFFFFF);i++);
printf("\n Job %d finished\n", counter);
pthread_mutex_unlock(&lock);
return NULL;
}
int main(void)
{
int i = 0;
int err;
if (pthread_mutex_init(&lock, NULL) != 0)
{ printf("\n mutex init failed\n");
return 1;
}
while(i < 2)
{
err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL);
if (err != 0)
printf("\ncan't create thread :[%s]", strerror(err));
i++;
}
pthread_join(tid[0], NULL);
pthread_join(tid[1], NULL);
pthread_mutex_destroy(&lock);
return 0;}
```

SAMPLE OUTPUT

```
    Job 1 started
Job 1 finished
Job 2 started
Job 2 finished
```

RESULT

Ex.No:12.a	**MEMORY ALLOCATION METHODS FOR FIXED PARTITION**
	FIRST FIT

PROGRAM

```
#include<stdio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - Worst Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1) //if bf[j] is not allocated
{
temp=b[j]-f[i];
if(temp>=0)
if(highest<temp)
{
ff[i]=j;
highest=temp;
}
}
}
frag[i]=highest;
bf[ff[i]]=1;
highest=0;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

SAMPLE OUTPUT

```
Enter the number of blocks:4
Enter the number of files:5
Enter the size of the blocks:-
Blocks 1:2
Blocks 2:8
Blocks 3:2
Blocks 4:1
Blocks 5:7
Enter the size of the files:-
File 1:1
File 2:8
File 3:2
File 4:1
File 5:7
File No File Size     Block No      Block Size      Fragment
1              1             1              2             1
2              8             2              8             0
3              2             3              6             4
4              1             4              4             6
```

RESULT

Ex.No:12.b	MEMORY ALLOCATION METHODS FOR FIXED PARTITION
	WORST FIT

PROGRAM:

```
#include<stdio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];

printf("\n\tMemory Management Scheme - First Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
{
ff[i]=j;
break;
}
}
}
frag[i]=temp;
bf[ff[i]]=1;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
```

```
    for(i=1;i<=nf;i++)
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

}
```

SAMPLE OUTPUT

```
        Memory Management Scheme-
    First FitEnter the number of blocks:3
    Enter the number of files:3
    Enter the size of the blocks:-
    Blocks 1:2
    Blocks 2:8
    Blocks 3:2
    Enter the size of the files:-
    File 1:1
    File 2:8
    File 3:2
    File No File Size       Block No       Block Size      Fragment
    1                1              1               2             1
    2                8              2               8             0
```

RESULT

Ex.No:12.c	**MEMORY ALLOCATION METHODS FOR FIXED PARTITION**
	BEST FIT

PROGRAM

```
#include<stdio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];

printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}

for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++) {
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
if(lowest>temp)
{
ff[i]=j;
lowest=temp;
}
}
}
frag[i]=lowest;
bf[ff[i]]=1;
lowest=10000;
}
printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

}
```

SAMPLE OUTPUT

```
        Memory Management Scheme-
    First FitEnter the number of blocks:3
    Enter the number of files:3
    Enter the size of the blocks:-
    Blocks 1:2
    Blocks 2:8
    Blocks 3:2
    Enter the size of the files:-
    File 1:1
    File 2:8
    File 3:2
    File No File Size       Block No      Block Size      Fragment
    1                1             1               2             1
    2                8             2               8             0
```

RESULT

Ex.No:13.a	**PAGE REPLACEMENT ALGORITHMS**
	FIFO

PROGRAM

```
#include<stdio.h>
int main()
{
int i=0,j=0,k=0,i1=0,m,n,rs[30],flag=1,p[30];
//system("clear");
printf("FIFO page replacement algorithm. .. \\n");
printf("enter the no. of frames:");
scanf("%d",&n);
printf("enter the reference string:");
while(1)
{
scanf("%d",&rs[i]);
if(rs[i]==0)
break;
i++;
}
m=i;
for(j=0;j<n;j++)
p[j]=0;
for(i=0;i<m;i++)
{
flag=1;
for(j=0;j<n;j++)
if(p[j]==rs[i]) {
printf("data already in page ...\n");
flag=0;
break;
}
if(flag==1)
{
p[i1]=rs[i];
i1++;
k++;
if(i1==n)
i1=0;
for(j=0;j<n;j++)
{
printf("\n page %d:%d",j+1,p[j]);
if(p[j]==rs[i])
printf("*");
}
printf("\n\n");
}
}
printf("total no page faults=%d",k);
}
```

SAMPLE OUTPUT

```
……………………………………………FIFO page replacement algorithm        \nenter the no. of frames:1
enter the reference string:2
panimalar

 page 1:2*          page 1:99*             page 1:206*    page 1:102*     page 1:1627410433*page
 1:1229148993*   page 1:1629618249*        page 1:4*                          page  1:1629200883*
 page 1:99*             page 1:1630529972*  page 1:1629950176*  page 1:102*
 page 1:1629422784*            page 1:102*    page 1:1630515492*     page 1:1627484735*page
 1:1630515584*                      page 1:1629467168*  data already in page....
 page 1:1629467250*
 page 1:1629450592*
 page 1:2*
 page 1:1630176576*
data already in page....
 page 1:1629626639*
 page 1:6411548*
 page 1:2*
 page 1:1628961560*
 page 1:1629450592*
 page 1:1*
 page 1:33*
 page 1:1*
total no page faults=31
```

RESULT

Ex.No:13.b	PAGE REPLACEMENT ALGORITHMS
	LRU

PROGRAM

```
#include<stdio.h>
void main()
{
int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
printf("Enter no of pages:");
scanf("%d",&n);
printf("Enter the reference string:");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
printf("Enter no of frames:");
scanf("%d",&f);
q[k]=p[k];
printf("\n\t%d\n",q[k]);
c++;
k++;
for(i=1;i<n;i++)
{
c1=0;
for(j=0;j<f;j++)
{
if(p[i]!=q[j])
c1++;
}
if(c1==f)
{
c++;
if(k<f)
{q[k]=p[i];
k++;
for(j=0;j<k;j++)
printf("\t%d",q[j]);
printf("\n");
}
else
{for(r=0;r<f;r++)
{c2[r]=0;
for(j=i-1;j<n;j--)
{if(q[r]!=p[j])
c2[r]++;
else
break;
}}
for(r=0;r<f;r++)
b[r]=c2[r];
for(r=0;r<f;r++)
{
for(j=r;j<f;j++)
{
if(b[r]<b[j])
{
t=b[r];
b[r]=b[j];
b[j]=t;
}}}
for(r=0;r<f;r++)
{
if(c2[r]==b[0])
q[r]=p[i];
printf("\t%d",q[r]);
}
printf("\n");
}}}
printf("\nThe no of page faults is %d",c);
}
```

SAMPLE OUTPUT

```
Enter no of pages:1
Enter the reference string: pani
Enter no of frames:
      1629091941

The no of page faults is 1
```

RESULT

Ex.No:13.c	PAGE REPLACEMENT ALGORITHMS
	LFU

PROGRAM

```
#include<stdio.h>
int main()
{
int f,p;
int pages[50],frame[10],hit=0,count[50],time[50];
int i,j,page,flag,least,minTime,temp;
printf("Enter no of frames : ");
scanf("%d",&f);
printf("Enter no of pages : ");
scanf("%d",&p);
for(i=0;i<f;i++)
{
frame[i]=-1;
}
for(i=0;i<50;i++)
{
count[i]=0;
}
printf("Enter page no : \n");
for(i=0;i<p;i++)
{
scanf("%d",&pages[i]);
}
printf("\n");
for(i=0;i<p;i++)
{
count[pages[i]]++;
time[pages[i]]=i;
flag=1;
least=frame[0];
for(j=0;j<f;j++)
{
if(frame[j]==-1 || frame[j]==pages[i])
{
if(frame[j]!=-1)
{
hit++;
}
flag=0;
frame[j]=pages[i];
break;
}
if(count[least]>count[frame[j]])
{
least=frame[j];
}
}
if(flag)
{
minTime=50;
for(j=0;j<f;j++)
{
if(count[frame[j]]==count[least] && time[frame[j]]<minTime)
{
temp=j;
minTime=time[frame[j]];
}
}
count[frame[temp]]=0;
frame[temp]=pages[i];
}
for(j=0;j<f;j++)
{
printf("%d ",frame[j]);
}
printf("\n");
}
printf("Page hit = %d",hit);
return 0;
}
```

SAMPLE OUTPUT

```
  Enter no of frames : 1

  Enter no of pages : 2

   Enter page no :
  1
  1
  1
  1
            Page hit = 1
```

RESULT

Ex.No:14.a	**FILE ORGANIZATION TECHNIQUE**
	SINGLE LEVEL DIRECTORY

AIM

To write C program to organize the file using single level directory.

ALGORITHM

Step1: Start the program.
Step2: Declare the count, file name, graphical interface.
Step3: Read the number of files
Step4: Read the file name
Step5: Declare the root directory
Step6: Using the file eclipse function define the files in a single level
Step7: Display the files
Step8: Stop the program

FLOWCHART

PROGRAM

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

#define MAX_FILES 100
#define MAX_FILENAME_LENGTH 50

void createFile(const char *filename);
void listFiles();
void deleteFile(const char *filename);

int main() {
    int choice;
    char filename[MAX_FILENAME_LENGTH];

    while (1) {
        printf("\nSingle-Level Directory Structure\n");
        printf("1. Create File\n");
        printf("2. List Files\n");
        printf("3. Delete File\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter filename to create: ");
                scanf("%s", filename);
                createFile(filename);
                break;
            case 2:
                listFiles();
                break;
            case 3:
                printf("Enter filename to delete: ");
                scanf("%s", filename);
                deleteFile(filename);
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void createFile(const char *filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error creating file.\n");
    } else {
        printf("File created successfully.\n");
        fclose(file);
    }
}

void listFiles() {
    printf("List of files in the directory:\n");
    system("ls -l");
}

void deleteFile(const char *filename) {
    if (remove(filename) == 0) {
        printf("File '%s' deleted successfully.\n", filename);
    } else {
        printf("Error deleting file '%s'.\n", filename);
    }
}
```

SAMPLE OUTPUT

```
   Single-Level Directory Structure
1. Create File
2. List Files
3. Delete File
4. Exit

Enter your choice: 1
Enter filename to create: panimalar
File created successfully.
1. Create File
2. List Files
3. Delete File
4. Exit
Enter your choice: 2

List of files in the directory:
total 755

-rwxr-xr-x    1 Administ UsersGrp       869 Feb  1 11:00 EX5C.C
-rwxr-xr-x    1 Administ UsersGrp     66793 Feb 1 11:21 a.exe
```

RESULT

Ex.No:14.b	**FILE ORGANIZATION TECHNIQUE**
	TWO LEVEL DIRECTORY

PROGRAM

```
    #include<string.h>
    #include<stdlib.h>
    #include<stdio.h>
    struct
    {
    char dname[10],fname[10][10];
    int fcnt;
    }dir[10];
    void main()
    {
    int i,ch,dcnt,k;
    char f[30], d[30];
    dcnt=0;
    while(1)
    {
    printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
    printf("\n4. Search File\t\t5. Display\t6. Exit\tEnter your choice -- ");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1: printf("\nEnter name of directory -- ");
    scanf("%s", dir[dcnt].dname);
    dir[dcnt].fcnt=0;
    dcnt++;
    printf("Directory created");
    break;
   case 2: printf("\nEnter name of the directory -- ");
   scanf("%s",d);
  for(i=0;i<dcnt;i++)
 if(strcmp(d,dir[i].dname)==0)
 {
printf("Enter name of the file -- ");
scanf("%s",dir[i].fname[dir[i].fcnt]);
printf("File created");
break;
}
if(i==dcnt)
printf("Directory %s not found",d);
break;
case 3: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++)
{
if(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is deleted ",f);
dir[i].fcnt--;
strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
goto jmp;
}
}
printf("File %s not found",f);
goto jmp;
}
}
printf("Directory %s not found",d);
jmp : break;
case 4: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter the name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++)
{
if(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is found ",f);
goto jmp1;
}
}
printf("File %s not found",f);
goto jmp1;
}
}
printf("Directory %s not found",d);
jmp1: break;
case 5: if(dcnt==0)
printf("\nNo Directory's ");
else
{
printf("\nDirectory\tFiles");
for(i=0;i<dcnt;i++)
{
printf("\n%s\t\t",dir[i].dname);
for(k=0;k<dir[i].fcnt;k++)
printf("\t%s",dir[i].fname[k]);
}
}
break;
default:exit(0);
```

```
}
}
}
```

SAMPLE OUTPUT

```
1.                  Create Directory    2. Create File  3. Delete File
4. Search File      5. Display    6. Exit Enter your choice -- 1
Enter name of directory -- panimalar
Directory created

1. Create Directory    2. Create File  3. Delete File
4. Search File      5. Display    6. Exit Enter your choice -- 2
Enter name of the directory -- panimalar
Directory panimalar not found

1. Create Directory    2. Create File  3. Delete File
4. Search File      5. Display    6. Exit Enter your choice -- 5
Directory     Files
panimalar
```

RESULT

Ex.No:15.a	FILE ALLOCATION STRATEGIES
	SEQUENTIAL

PROGRAM

```
#include<stdio.h>
struct student
{int sno;
char name[25];
int m1,m2,m3;
}s;
void display(FILE *);
int search(FILE *,int);
void main()
{int i,n,sno_key,opn;
FILE *fp;
//struct student s;
printf("How many records ?");
scanf("%d",&n);
fp=fopen("stud.dat","w");
for(i=0;i<n;i++)
{printf("Enter the student information : %d(sno,Name,M1,M2,M3):",i+1);
scanf("%d%s%d%d%d",&s.sno,s.name,&s.m1,&s.m2,&s.m3);
fwrite(&s,sizeof(s),1,fp);
}
fclose(fp);
fp=fopen("stud.dat","r");
do
{printf("1-DISPLAY\n2.SEARCH\n 3.EXIT\n YOUR OPTION: ");
scanf("%d",&opn);
switch(opn)
{
case 1:
printf("\n Student Records in the file \n");
display(fp);
break;
case 2:
printf("Read sno of the student to be searched :");
scanf("%d",&sno_key);
if(search(fp,sno_key)){
printf("success!! Record found in the file\n");
printf("%d\t%s\t%d\t%d\t%d\n", s.sno,s.name,s.m1,s.m2,s.m3);
}
else
printf("Failure!! Record %d not found\n",sno_key);
break;
case 3:
printf("Exit !! press key");
break;
default:
printf("Invalid option!!! Try again!!\n");
break;
}
}while(opn!=3);
fclose(fp);
}
void display(FILE *fp)
{rewind(fp);
while(fread(&s,sizeof(s),1,fp))
printf("%d\t%s\t%d\t%d\t%d\n",s.sno,s.name,s.m1,s.m2,s.m3);
}
int search(FILE *fp,int sno_key)
{rewind(fp);
while(fread(&s,sizeof(s),1,fp))
if(s.sno==sno_key)
return 1;
return 0;
}
```

SAMPLE OUTPUT

```
        How many records ?2
           Enter the student information :
        1(sno,Name,M1,M2,M3):1stud1    54      52
           154
  Enter the student information : 2(sno,Name,M1,M2,M3):
 2
           stud2   52      4    1
1-DISPLAY
2.SEARCH
 3.EXIT
 YOUR OPTION: 1

 Student Records in the file
1      stud1   54     52     154
      2              stud2   52  4        1
1-DISPLAY
2.SEARCH
 3.EXIT
 YOUR OPTION: 3
Exit !! press key
```

RESULT

Ex.No:15.b	FILE ALLOCATION STRATEGIES
	LINKED

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
struct emp {
    char name[50];
    float salary;
    int age;
    int id;
    };
    struct emp e;
    long int size = sizeof(e);
    COORD cord = { 0, 0 };
    void gotoxy(int x, int y)
    {
    cord.X = x;
    cord.Y = y;
    SetConsoleCursorPosition( GetStdHandle(STD_OUTPUT_HANDLE), cord);
    }
    FILE *fp, *ft;
    void addrecord()
    {
    system("cls");
    fseek(fp, 0, SEEK_END);
    char another = 'y';
    while (another == 'y') {
            printf("\nEnter Name : ");
            scanf("%s", e.name);
     printf("\nEnter Age : ");
     scanf("%d", &e.age);
     printf("\nEnter Salary : ");
     scanf("%f", &e.salary);
     printf("\nEnter EMP-ID : ");
     scanf("%d", &e.id); fwrite(&e,
     size, 1, fp);
            printf("\nWant to add another" " record (Y/N) : ");
            fflush(stdin);
            scanf("%c", &another);
    }
    }
    void deleterecord()
    {
    system("cls");
    char empname[50];
    char another = 'y';
    while (another == 'y') {
    printf("\nEnter employee ""name to delete : ");
    scanf("%s", empname);
```

```
        ft = fopen("temp.txt", "wb");
        rewind(fp);
        while (fread(&e, size, 1, fp) == 1) {
            if (strcmp(e.name, empname) != 0)
                fwrite(&e, size, 1, ft);
        }
        fclose(fp);
        fclose(ft);
        remove("data.txt");
        rename("temp.txt", "data.txt");
        fp = fopen("data.txt", "rb+");
        printf("\nWant to delete another"
            " record (Y/N) :");
        fflush(stdin);
    }
}
    void displayrecord()
        {
    system("cls");
    rewind(fp);
    printf("\n========================="
        "==========================="
        "======");
    printf("\nNAME\t\tAGE\t\tSALARY\t\t"
        "\tID\n",
        e.name, e.age,
        e.salary, e.id);
    printf("==========================="
        "==========================="
        "====\n");
    while (fread(&e, size, 1, fp) == 1)
        printf("\n%s\t\t%d\t\t%.2f\t%10d",
        e.name, e.age, e.salary, e.id);
        printf("\n\n\n\t");
    system("pause");
   }
    void modifyrecord()
    {
    system("cls");
    char empname[50];
    char another = 'y';
    while (another == 'y') {
        printf("\nEnter employee name"
            " to modify : ");
        scanf("%s", empname);
        rewind(fp);
        while (fread(&e, size, 1, fp) == 1) {
            if (strcmp(e.name, empname) == 0) {
                printf("\nEnter new name:");
                scanf("%s", e.name);
                printf("\nEnter new age :");
                scanf("%d", &e.age);
                printf("\nEnter new salary :");
```

```
                                        scanf("%f", &e.salary);
                                        printf("\nEnter new EMP-ID :");
                                        scanf("%d", &e.id);
                                        fseek(fp, -size, SEEK_CUR);
                                        fwrite(&e, size, 1, fp);
                                        break;
                                }
                        }
                        printf("\nWant to modify another"
                                " record (Y/N) :");
                        fflush(stdin);
                        scanf("%c", &another);
                }
        }
                int main()
                {
                int choice;
                fp = fopen("data.txt", "rb+");
                if (fp == NULL) {
                        fp = fopen("data.txt", "wb+");
                        if (fp == NULL) {
                                printf("\nCannot open file...");
                                exit(1);
                        }
                }
                system("Color 3F");
                printf("\n\n\n\n\t\t\t\t============="
                        "============================="
                        "===========");
                printf("\n\t\t\t\t~~~~~~~~~~~~~~~~~~~"
                        "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
                        "~~~~~");
                printf("\n\t\t\t\t==================="
                        "============================="
                        ""=====");
                printf("\n\t\t\t\t[|:::>:::>:::>::> "
                        "EMPLOYEE RECORD <::<:::<:::"
                        "<:::|]\t");
                printf("\n\t\t\t\t==================="
                        "============================="
                        "=====");
                printf("\n\t\t\t\t~~~~~~~~~~~~~~~~~~~~"
                        "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
                        "~~~");
                printf("\n\t\t\t\t====================="
                        "==============================\n");
                printf("\n\n\n\t\t\t\t\t\t\t\t\t\t"
                        "Developer : @Sushant_Gaurav"
                        "\n\n\t\t\t\t");
                system("pause");
                while (1) {
                        system("cls");
                        gotoxy(30, 10);
```

```
printf("\n1. ADD RECORD\n");
gotoxy(30, 12);
printf("\n2. DELETE RECORD\n");
gotoxy(30, 14);
printf("\n3. DISPLAY RECORDS\n");
gotoxy(30, 16);
printf("\n4. MODIFY RECORD\n");
gotoxy(30, 18);
printf("\n5. EXIT\n");
gotoxy(30, 20);
printf("\nENTER YOUR CHOICE...\n");
fflush(stdin);
scanf("%d", &choice);
switch (choice) {
case 1:
        addrecord();
        break;
case 2:
        deleterecord();
        break;
case 3:
        displayrecord();
        break;
case 4:
        modifyrecord();
        break;
case 5:
        fclose(fp);
        exit(0);
        break;
default:
        printf("\nINVALID CHOICE...\n");
                               }
                       }
                       return 0;
               }
```

SAMPLE OUTPUT

```
 1. ADD RECORD
 2. DELETE RECORD
 3. DISPLAY RECORDS
 4. MODIFY RECORD

 5. EXIT

 ENTER YOUR CHOICE...1
 sh: cls: command not found

 Enter Name : 3

 Enter Age : 54

 Enter Salary : 455525

 Enter EMP-ID : shd44
1. ADD RECORD

2. DELETE RECORD

3. DISPLAY RECORDS

4. MODIFY RECORD

5. EXIT

ENTER YOUR CHOICE...
3
sh: cls: command not found

===============================================================
NAME           AGE            SALARY                ID
===============================================================

3              54             455525.00             0
```

RESULT

Ex.No:15.c	**FILE ALLOCATION STRATEGIES**
	INDEXED

PROGRAM

```
#include  <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_EMPLOYEES 100
struct Employee {
       int id;
      char name[50];
      float salary;
      };
      void addEmployee(FILE *file);
      void searchEmployee(FILE *file, int id);
      void displayAllEmployees(FILE *file);
      int main() {
      FILE *file;
      struct Employee employees[MAX_EMPLOYEES];
      file = fopen("employees.dat", "rb+");
      if (file == NULL) {
       printf("File doesn't exist. Creating a new file...\n");
       file = fopen("employees.dat", "wb+");
       if (file == NULL) {
       printf("Error creating file. Exiting...\n");
       return 1;
       }
      }
        int choice;
       do {
       printf("\nEmployee Database\n");
       printf("1. Add Employee\n");
       printf("2. Search Employee\n");
       printf("3. Display All Employees\n");
        printf("4. Exit\n");
printf("Enter     your choice: ");
scanf("%d", &choice);
      switch (choice) {
         case 1:
            addEmployee(file);
            break;
         case 2: {
            int id;
            printf("Enter employee ID to search: ");
            scanf("%d",  &id);
            searchEmployee(file, id);
            break;
         }
         case 3:
                displayAllEmployees(file);
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);
    fclose(file);
    return 0;
}
             void addEmployee(FILE *file) {
             struct Employee employee;
             printf("Enter employee ID: ");
             scanf("%d", &employee.id);
             printf("Enter employee name: ");
             scanf("%s", employee.name);
             printf("Enter employee salary: ");
             scanf("%f", &employee.salary);
             fseek(file, (employee.id - 1) * sizeof(struct Employee), SEEK_SET);
             fwrite(&employee, sizeof(struct Employee), 1, file);
             printf("Employee added successfully.\n");
}
             void searchEmployee(FILE *file, int id) {
             struct Employee employee;
             fseek(file, (id - 1) * sizeof(struct Employee), SEEK_SET);
             fread(&employee, sizeof(struct Employee), 1, file);
              if (employee.id == 0) {
              printf("Employee with ID %d not found.\n", id);
    }        else {
              printf("Employee details:\n");
              printf("ID: %d\n", employee.id);
              printf("Name: %s\n", employee.name);
             printf("Salary: %.2f\n", employee.salary);
    }
}
void displayAllEmployees(FILE *file) {
struct Employee employee;
rewind(file);
    printf("All Employees:\n");
    while (fread(&employee, sizeof(struct Employee), 1, file) == 1) {
        if (employee.id != 0) {
            printf("ID: %d, Name: %s, Salary: %.2f\n", employee.id, employee.name, employee.salary);
        }
    }
}
```

SAMPLE OUTPUT

```
 File doesn't exist. Creating a new file...

Employee Database
1. Add Employee
2. Search Employee
3. Display All Employees
4. Exit
Enter your choice: 1
Enter employee ID: 5
Enter employee name: emp1
Enter employee salary: 55654
Employee added successfully.
Employee Database
1. Add Employee
2. Search Employee
3. Display All Employees
4. Exit
Enter your choice: 3
All Employees:
ID: 5, Name: emp1, Salary: 55654.00
 Employee Database
1. Add Employee
2. Search Employee
3. Display All Employees
4. Exit
Enter your choice: 4
Exiting...
```

RESULT

Ex.No:16 a	DISK SCHEDULING ALGORITHMS
	FCFS

PROGRAM

```
    #include <stdio.h>
    #include <stdlib.h>
    void fcfs(int arr[], int head, int size)
    {
     int total_movement = 0;
     printf("Sequence of disk accesses:\n");
     for (int i = 0; i < size; i++)
   {
      int distance = abs(head - arr[i]);
      total_movement += distance;
      printf("%d ", arr[i]);
      head = arr[i];
    }
  printf("\nTotal head movement: %d\n", total_movement);
}
   int main()
{
  int n, head;
  printf("Enter the number of requests: ");
  scanf("%d", &n);
  int arr[n];
  printf("Enter the requests:\n");
  for (int i = 0; i < n; i++)
  {
     scanf("%d", &arr[i]);
  }
  printf("Enter the initial position of the head: ");
  scanf("%d", &head);
  fcfs(arr, head, n);
  return 0;
}
```

SAMPLE OUTPUT

```
        Enter the number of requests: 2
    Enter the requests: 12
    Enter the initial position of the head: 4
    Sequence of disk accesses:

    1 2
    Total head movement: 4
```

Ex.No:16 b	**DISK SCHEDULING ALGORITHMS**
	SSTF

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
int findShortestSeekTime(int request_queue[], int head, int n)
{
   int distance, min_distance = 5;
   int i,index = -1;
   for (i = 0; i < n; i++)
   {
    distance = abs(head - request_queue[i]);
    if (distance < min_distance) {
       min_distance = distance;
       index = i;
    }
   }
   return index;
}
void sstf(int request_queue[], int head, int n)
{
   int i,index, total_movement = 0;

   printf("Sequence of disk accesses:\n");
   while (n > 0)
   {
    index = findShortestSeekTime(request_queue, head, n);
    total_movement += abs(head - request_queue[index]);
    head = request_queue[index];
    printf("%d ", request_queue[index]);
    for (i = index; i < n - 1; i++)
    {
       request_queue[i] = request_queue[i + 1];
    }
    n--;
   }
   printf("\nTotal head movement: %d\n", total_movement);
}
int main()
{
   int n,i,head;
   int request_queue[10];
   printf("Enter the number of requests: ");
   scanf("%d", &n);
 printf("Enter the requests:\n");
```

```
    for (i = 0; i < n; i++)
    {
      scanf("%d", &request_queue[i]);
    }
    printf("Enter the initial position of the head: ");
    scanf("%d", &head);
    sstf(request_queue, head, n);
    return 0;
}
```

SAMPLE OUTPUT

```
Enter the number of requests: 2
Enter the requests:
1
2
Enter the initial position of the head: 4
Sequence of disk accesses:
2 1
Total head movement: 3
```

RESULT

Ex.No:16 c	DISK SCHEDULING ALGORITHMS
	SCAN

AIM

PROGRAM

```
  #include <stdio.h>
  #include <stdlib.h>
  void sort(int arr[], int n)
  {
  int i,j;
       for (i = 0; i < n; i++)
       {
        for (j = i+1; j < n; j++)
       {
       if (arr[i] > arr[j])
       {
          int temp = arr[i];
          arr[i] = arr[j];
          arr[j] = temp;
       }      }    }}
       void scan(int request_queue[], int head, int n, int max)
       {  int i,j;
       int total_movement = 0;
        int direction = 1; // 1 represents right, -1 represents left
        printf("Sequence of disk accesses:\n");
       while (n > 0)
       {
   sort(request_queue, n);
   for (i = 0; i < n; i++)
   {
       if ((direction == 1 && request_queue[i] >= head) ||
          (direction == -1 && request_queue[i] <= head))
       {
          total_movement += abs(head - request_queue[i]);
          head = request_queue[i];
          printf("%d ", request_queue[i]);
          for (j = i; j < n - 1; j++)
          {
             request_queue[j] = request_queue[j + 1];
          }
          n--;
          i--;
       }}

if (n > 0)
{
```

```
            direction = -direction;
    }  }
    printf("\nTotal head movement: %d\n", total_movement);
}
int main()
{
    int n, head, max;
    printf("Enter the number of requests: ");
    scanf("%d", &n);
    int request_queue[10];
    printf("Enter the requests:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &request_queue[i]);
    }
    printf("Enter the initial position of the head: ");
    scanf("%d", &head);
    printf("Enter the maximum track number: ");
    scanf("%d", &max);
    scan(request_queue, head, n, max);
    return 0;
}
```

SAMPLE OUTPUT

```
Enter the number of requests: 2
Enter the requests:
1
3
Enter the initial position of the head: 4
Enter the maximum track number: 2
Sequence of disk accesses:
1 3
Total head movement: 5
```

RESULT

Ex.No:16 d	DISK SCHEDULING ALGORITHMS
	C-SCAN

AIM

To write a C program for implementation of C - SCAN Disk Scheduling Algorithm (C-SCAN).

ALGORITHM

```
Step 1: Start
Step 2:Read n (number of requests).
         Read head (initial position of the disk head).
         Read request_queue (an array of n disk requests).
Step 3: Initialize:
          Set total_movement to 0.
          Set the direction to "right."
Step 4: Process Requests:
              While there are remaining requests:
              Sort the request queue based on the distance from the current head position.
              Scan the sorted request queue in the current direction.
              Update total_movement by adding the seek time for each serviced request.
              Move the head to the last serviced request position.
              Remove the serviced requests from the queue.
              If there are remaining requests, change the direction.
              If the head reaches the maximum track number, move it to the beginning.
Step 5: Print "Sequence of disk accesses:" and the sequence of serviced requests.
          Print "Total head movement: total_movement".
Step 6: Stop
```

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
void sort(int arr[], int n)
{ int i,j;
     for (i = 0; i < n; i++)
    {
 for (j = i+1; j < n; j++)
 {if (arr[i] > arr[j])
    {
       int temp = arr[i];
       arr[i] = arr[j];
       arr[j] = temp;
    }       }    }}
    void cscan(int request_queue[], int head, int n, int max)
    {
    int i,j;
```

```
              int total_movement = 0;
             int direction = 1; // 1 represents right, -1 represents left
    printf("Sequence of disk accesses:\n");
    while (n > 0)
    {
        sort(request_queue, n);
            for ( i = 0; i < n; i++)
        {
            if ((direction == 1 && request_queue[i] >= head && request_queue[i] <= max) ||
                (direction == -1 && request_queue[i] >= 0 && request_queue[i] <= head))
                {
                total_movement += abs(head - request_queue[i]);
                head = request_queue[i];
                printf("%d ", request_queue[i]);
                        for (j = i; j < n - 1; j++)
                {
                    request_queue[j] = request_queue[j + 1];
                }
                n--;
                i--; // Adjust index since we removed an element
             } }
        if (n > 0)
        {
            direction = -direction;
        }
    if (head == max)
        {
            head = 0;
        }  }
    printf("\nTotal head movement: %d\n", total_movement);
}
int main()
{
    int n, head, max;
    int request_queue[100];
    printf("Enter the number of requests: ");
    scanf("%d", &n);
    printf("Enter the requests:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &request_queue[i]);
    }
    printf("Enter the initial position of the head: ");
    scanf("%d", &head);
    printf("Enter the maximum track number: ");
    scanf("%d", &max);
    cscan(request_queue, head, n, max);
return 0;}
```

SAMPLE OUTPUT

```
Enter the number of requests: 2
Enter the requests:
1
3
Enter the initial position of the head: 4
Enter the maximum track number: 2
Sequence of disk accesses:
1 3
Total head movement: 5
```

RESULT

Ex. No: 17	INSTALL ANY GUEST OPERATING SYSTEM LIKE LINUX USING VMWARE

AIM

To install any guest operating system link linux using VMWare.

PROCEDURE

To create a virtual machine in vCenter Server for each remote desktop that is deployed in a Horizon 8 environment. You must install your Linux distribution on the virtual machine.

Prerequisites

- Verify that your deployment meets the requirements for supporting Linux desktops. See System Requirements for Horizon Agent for Linux.
- Familiarize yourself with the steps for creating virtual machines in vCenter Server and installing guest operating systems. For more information see the *Windows Desktops and Applications in Horizon* document.
- Familiarize yourself with the video memory (vRAM) settings requirements for the monitors you plan to use with the virtual machine. See System Requirements for Horizon Agent for Linux.

1. In vSphere Client, create a virtual machine.
2. Configure custom configuration options.
   a. Right-click the virtual machine and click Edit Settings.
   b. Specify the number of vCPUs and the vMemory size. For the required settings, refer to the following guidelines.
      - If you are preparing the virtual machine for deployment as a single-session virtual desktop pool, follow the guidelines in the installation guide for your Linux distribution.

        For example, Ubuntu 18.04 specifies configuring 2048 MB for vMemory and 2 vCPUs.
      - If you are preparing the virtual machine to serve as a multi-session host for a published desktop or application pool, specify at least 8 vCPUs and 40 GB of vMemory.

3. Power on the virtual machine and install the required Linux distribution. Note the following considerations for instant-clone desktop pools and multi-session hosts.

   Horizon Agent for Linux only supports instant-clone desktop pools created from virtual machines running the following operating systems:

   a. Ubuntu 18.04/20.04/22.04
   b. RHEL 7.x/8.x/9.x
   c. CentOS 7.8/7.9
   d. SLED/SLES 12.x/15.x

   Only virtual machines running RHEL Workstation 7.8 or later, RHEL Workstation 8.1 or later, RHEL Workstation 9.0 or later, or Ubuntu 18.04/20.04/22.04 can support multi-session published desktop pools and single-session or multi-session application pools.

4. Configure the desktop environment to use for the specific Linux distribution.

   See the Desktop Environment section in System Requirements for Horizon Agent for Linux for additional information.

5. Ensure that the system hostname is resolvable to 127.0.0.1

RESULT

Thus guest operating system linux using VMWare has been successfully installed.