1] STUDENT DATABASE(SHELL)

```
clear
while [ 1 ]
do
  echo 1.Create 2.Display 3.Insert 4.Search 5.Modify 6.Delete 7.Exit
  echo “Enter choice=\c”
  read $ch
  case $ch in
  1)    echo -n “Enter the file name”
        read $fname
        if [ -e $fname ]
        then
               echo “File already exists”
        else
              >> $fname
              echo “File created successfully”
        fi
        ;;
  2)    echo -n “Enter the file name”
        read $fname
        if [ -e $fname ]
        then
        echo “File content:”
        sort -n $fname
        else
        echo “File dne”
        fi
        ;;
  3)    echo -n “Enter the file name”
        read $fname
        if [ -e $fname ]
        then
        echo “Enter the roll number”
        read $roll
        grep -w “$roll” $fname
        ans=$?
        if [ ans -eq 0 ]
        then echo “Record already exists”
        else
              echo “Enter the name”
              read $name
              echo $roll $name >> $fname
              echo “Record inserted successfully”
        fi
        else echo “File DNE”
        fi
```

```
      ;;
4)    echo -n “Enter the file name”
      read $fname
      if [ -e $fname ]
      then echo “Enter the roll number”
              read $roll
              grep -w “$roll” $fname
              ans=$?
              if [ ans -eq 0 ]
              then echo “Record found”
              else echo “Record not found”
              fi
      else echo “File DNE”
      fid
      ;;
5)    echo -n “Enter the file name”
      read $fname
      if [ -e $fname ]
      then echo “Enter the roll number”
             read $roll
             grep -w “$roll” $fname
             ans=$?
             if [ ans -eq 0 ]
             then echo “Enter newroll newname”
                    read $nroll $nname
                    grep -w “$nroll” $fname
                    ans=$?
                    if [ ans -eq 0 ]
                    then echo “Record already exists”
                    else grep -v “$roll” $fname >> temp
                          echo nroll nname >> temp
                          rm $fname
                          cp temp $fname
                          rm temp
                          echo “Record modified successfully”
                    fi
              else echo “Record not found”
              fi
       else echo “File DNE”
       fi
       ;;
6)    echo -n “Enter the file name”
      read $fname
      if [ -e $fname ]
      then echo -n “Enter the roll number”
      read $roll
      grep -w “$roll” $fname
      ans=$?
```

```
        if [ ans -eq 0 ]
        then grep -v “$roll” $fname >> temp
                rm $fname
                cp temp $fname
                rm temp
                echo “Record deleted successfully”
        else echo “Record does not exist”
        fi
        else echo “File not found:
        fi
        ;;
   7)   exit
        ;;
   *)   echo “Wrong choice”
        ;;
esac
done
```

2]PROCESS CREATION(C)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/wait.h>

void asc(int *,int)
void desc(int *,int)

int main(){
int *a,n,i;
pid_t pid;
printf("Enter the number of array elements:");
scanf("%d",&n);
a=(int *)malloc(n*sizeof(int));
printf("Enter the array elements:");
for(i=0;i<n;i++){
printf("\na[%d]: ",i);
scanf("%d",&a[i]);
}
printf("\n");

pid=fork();

if(pid<0)
{
perror("Fork error\n");
}
else if(pid==0){printf("Child process id : %ld",(long)getpid());}
else{printf("Parent process id : %ld",(long)getpid());}

switch(pid){
case -1: printf("\nFork error");
            exit(-1);
case 0: printf("Child process executes");
           asc(a,n);
           system("ps -elf");
          exit(pid);
default: wait(NULL);
            printf("Parent process executes");
           desc(a,n);
           // system("ps -elf");
          exit(pid);
}
}
```

```
void asc(int *a,int n){
int i,j,temp;
for(i=0;i<n;i++){
for(j=0;j<n-1;j++){
if(a[j]>a[j+1]){
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
}
void desc(int *a,int n){
int i,j,temp;
for(i=0;i<n;i++){
for(j=0;j<n-1;j++){
if(a[j]<a[j+1]){
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
}
```

3] FCFS SCHEDULING ALGORITHM(C)

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

struct proc{ int proc,at,bt,tat,wt; }
void sort();
void calculate();
int temp,a,n,b;
struct proc p[100];

int main(){
printf("Enter the number of processes");
scanf("%d",&n);
printf("\n");
for(int i=0;i<n;i++){
scanf("%d %d %d", &p[i].proc, &p[i].at, &p[i].bt);
}
printf("\n Process id \t Arrival time \t Burst time");
for(int i=0;i<n;i++){
printf("%d \t\t %d \t\t %d",p[i].proc, p[i].at, p[i].bt);
printf("\n");
}
sort();
calculate();
}

void sort(){
int i,j;
struct proc temp;
for(int i=0;i<n;i++){
for(int j=0;j<n-1;j++){
if(p[j].at>p[j+1].at){
temp=p[j];
p[j]=p[j+1];
p[j+1]=temp;
}
}
}
printf("Processes in sorted order:\n");
printf("\n Process id \t Arrival time \t Burst time");
for(int i=0;i<n;i++){
printf("%d \t\t %d \t\t %d",p[i].proc, p[i].at, p[i].bt);
printf("\n");
}
}


void calculate(){
int i;
float atat,awt,a=0,b=0;

p[0].tat=p[0].at+p[0].bt;

for(i=1;i<n;i++){
if(p[i-1].tat>=p[i].at){p[i].tat=p[i-1].tat+p[i].bt;}
else{p[i].tat=p[i].at+p[i].bt;}

for(i=0;i<n;i++){
p[i].tat=p[i].tat-p[i].at;
p[i].wt=p[i].tat-p[i].bt;
a=a+p[i].tat;
b=b+p[i].wt;
}

atat=a/n;
awt=b/n;
printf("\n");
printf("Process id \t Arrival Time \t Burst time \t Turnaround Time \t Waiting Time \n");
for(i=0;i<n;i++){
printf("%d \t\t %d \t\t %d \t\t\t %d \t\t %d",p[i].proc, p[i].at, p[i].bt, p[i].tat, p[i].wt);
printf("\n");
}
printf("Average turnaround time: %f,Average waiting time: %f",atat,awt");
}
```

4] ROUNDROBIN

5] READER WRITER

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

#define R 5
#define W 5
int readcount;
pthread_mutex_t x;
sem_t wsem;
int h=11,m=55;

void *reader1(void *a);
void *writer1(void *a);

int main(){
int i;
pthread_t thread_write[W],thread_read[R];
pthread_mutex_init(&x,NULL);
sem_init(&wsem,0,1);
printf("Readers have priority: ");
readcount=0;
for(i=0;i<W;i++){
pthread_create(&thread_write[i],NULL, *writer1, (void *)i);
}
for(i=0;i<R;i++){
pthread_create(&thread_read[i],NULL, *reader1,(void *)i);
}
for(i=0;i<W;i++){
pthread_join(&thread_write[i],NULL);
}
for(j=0;j<W;j++){
pthread_join(&thread_read[i],NULL);
}
}

void *reader1(void *a){
int r=(int) a;
int i=0;

while(i<5){
pthread_mutex_lock(&x);
readcount++;
if(readcount==1)
sem_wait(&wsem);
pthread_mutex_unlock(&x);
printf("Reader %d is reading : %d \t %d \n",r,h,m);
pthread_mutex_lock(&x);
readcount--;
if(readcount==0)
sem_post(&wsem);
pthread_mutex_unlock(&x);
sleep(rand() % 10);
i++;
}
}

void *writer1(void *a){
int w=(int) a;
int i=0;

while(i<2){
sem_wait(&wsem);
m=m+1;
if(m==60){
h=h+1;
m=0;
}
printf("Writer %d is writing: %d \t %d \n",w,h,m);
sem_post(&wsem);
sleep(rand() % 10);
i++;
}
}
```

For execution:
```
gcc rw.c -lpthread
./a.out
```

6] FULL DUPLEX COMMUNICATION(PRODUCER-CONSUMER)

7] FIFO PAGE REPLACEMENT

```
#include <stdio.h>

void main(){
int input[100],pages,frame_size,flag,i,j,page_hit=0,page_fault=0;
double fault_frequency,hit_ratio;

printf(“Enter the number of pages”);
scanf(“%d”,&pages);
scanf(“Enter the frame size”);
scanf(“%d”,&frame_size);

int queue[frame_size];
int f=0;

printf(“Enter the reference string”);
for(i=0;i<pages;i++){scanf(“%d”,&input[i]);}
for(i=0;i<frame_size;i++){queue[i]=999;}

for(i=0;i<pages;i++){
flag=0;
for(j=0;j<frame_size;j++){
if(queue[j]=input[i]){
flag=1;
page_hit++;
}
}
if(flag==0){
queue[f%frame_size]=input[i];
f++;
page_fault++;
}
for(j=0;j<frame_size;j++){
printf(“%d “,queue[i]);
}
printf(“\n”);
}
printf(“Page_faults: %d,Page_Hits: %d”,page_fault,page_hit);
fault_frequency=((double)page_fault/pages)*100;
hit_ratio=((double)page_hit/pages)*100;
}
```