week 1(a):

BIT STUFFING

```
#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
```

Week1(b):

CHAR STUFFING:

```
#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

Week2:

CRC:

```
#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

Week3:

STOP AND WAIT PROTOCOL:

```
#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

Week4:

SLIDING WINDOW PROTOCOL:

```
#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;
}
```

Week5:

SHORTEST PATH ALGORITHM:

```
#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.
```

Week 6:

BROADCAST TREE:

```
#include<stdio.h>
int a[10][10],n;
void adj(int k);
int 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);
return 0;
}
void 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);
}
}
```

Week7:

COLLISION FREE PROTOCOL:

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,k;
int sn;
int st[20];
printf("\n How many stations: ");
scanf("%d",&sn);
char op;
do
{
```

```
printf("\n Enter status of stations");
for (i=0;i<sn;i++)
{
printf("\n Enter status of station %d" :i+1 )";
scanf("%d",&st[i]);
}
//Print ready stations
for (i=0;i<sn;i++)
{
if(st[i]==1)
{
printf("\n Station %d is ready to transmit",i+1);
}
}
printf("\n Repeat? Press Y :");
scanf("%c",&op);
}
while(op=='y' || op=='Y');
getch();
}
```

OS PROGRAMS:

WEEK 9(a):

```
echo enter a number:
read num
fact=1
while[$num -gt 1]
```

```
do
fact=$((fact*num))
num=$((num-1))
done
echo factorial=$fact
```

Week 9(b):

```
#include<stdio.h>
#include<conio.h>
#include<sys/types.h>
int main()
{
pid_t pid=fork();
if(pid<0)
{
perror("fork failed");
}
if(pid==0){
printf("Child process is created\n");
}
else
{
printf("Parent process is created");
return 0;
}
}
```

Week 9(c):

```
#inlude<stdio.h>
#include<unistd.h>
int main()
{
int pipefdsp[2];
int returnstatus;
int pid;
char writemessage[2][20]={"Hi","Hello"};
char readmessage[20];
returnstatus=pipe(pipefds);
if(returnstatus==-1){
  printf("Unable to create ppe\n");
  return 1;
}
pid=fork();
if(pid==0){
   read(pipefds[0],readmessage,szeofreadmessage));
   printf("Child Process-Reading from pipe-Message 1 is %s\n",readmessage);
   read(pipefds[0],readmessage,szeofreadmessage));
   printf("Child Process-Reading from pipe-Message 2 is %s\n",readmessage);
}
else{
  printf("Parent process-Writing to pipe-message 1 is %s\n",writemessage[0]);
```

```
  write(pipefds[1],writemessages[0],sizeof(writemessage[0]));
  printf("Parent process-Writing to pipe-message 2 is %s\n",writemessage[1]);
  write(pipefds[1],writemessages[1],sizeof(writemessage[1]));
}
return 0;
}
```

Week 10(a):

FCFS:

```
#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
2
3
4
13
25
0
```

4

17

4

17

42

Avg.waiting time=7.00000

Avg turnaround time=21,00000

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

Week10(b):

PRIORITY:

```
#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);
}
```

Week 11(a):

SJF:

```
#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);
}
```

Week 11(b):

ROUND ROBIN:

```
#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);
}
```

Week 12(a):

SEQUENCIAL

```
#include<stdio.h>
void 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]);
}
```

Week12(b)

LINKED:

```
#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]);
}}
```

Week 12(c):

INDEXED

```
#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]);
}
```

WEEK 13(a):

PAGING:

```
#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;
for(i=0;i<pages;i++){
pos:
/* Initializing array elements with -1*/
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);
}
```

WEEK 13(b):

SEGMENTATION:

```
#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]);
}
}}
```

WEEK 14(a):

FIFO:

```
#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);
   }
```

WEEK 14(b):

LRU:

```
#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);
}
```

WEEK 14(c):

OPTIMAL:

```
#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;
}
```