WEEK 1A
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(); }
```

WEEK 1B
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\nDLESTX%sDLEETX ",b,c);
 return 0;}
```

WEEK 4
SLIDING WINDOW PROTOCOL

```
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;  }
```

WEEK 5
Dijkstra's SHORTEST PATH ALGO

```
#include <stdio.h>
#define infinity 9999
#define MAX 20
int minimum(int a, int b) {
   if (a <= b)      return a;
   else      return b;}
int 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]);     return 0;}
```

WEEK 3
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(); }
```

WEEK 2
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();  }
```

```
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
```

WEEK 9A
FACT OF AN INTEGER

```
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);    }
```

WEEK 9B
CREATING CHILD PROCESS

```
#include <stdio.h>
#include<sys/wait.h>
int main(void){
int pid;
int status;
printf("Hello World!\n");
pid = fork( );
if(pid == -1) { perror("bad fork");    exit(1);    }
if (pid == 0)     printf("I am the child process.\n");
else { wait(&status);
printf("I am the parent process.\n");
return 0; } }
```

WEEK 9C
PARENT WRITES MSG

```
#include<stdio.h>
int main()
{
int a[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 \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);
printf("\nAvg. Turn around time=%f",atat);  }
```

WEEK 10 FCFS

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

WEEK 10 PRIORITY

```
void main() {
int
id[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;   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 SHORTEST JOB FIRST

```
#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 11B 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 SEQUENTIAL FILE ALLOC

```
#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]);  }
```

WEEK 12 B LINKED ALGO

```
#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

```
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,pagesizepr
osize,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;  frames = memsize/pagesize;
printf("\nThe process is divided into %d pages",pages);
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: printf("\nEnter the frame number of page %d: ",i);
scanf("%d",&temp);
if(temp>=frames) {  printf("\n\t****Invalid frame number****\n");
goto pos; }
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;
temp = base + (temp*pagesize)+j;
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 PAGE REPLACE

```
#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;
if (avail==0)  {
frame[j]=a[i];
pagefault++;
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); }
```