PROGRAM:-01

P1 #include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

Int main() { Pid_t child_pid;

Child_pid = fork();

If (child_pid < 0) {

Perror("Fork failed"); Return 1;}

If (child_pid == 0) {

Printf("Child process (PID %d)

is executing.\n", getpid());

Char *args[] = {"/bin/ls", "-l", NULL};

Execv(args[0], args);

Perror("Exec failed");

Return 1; } else {

Printf("Paent process (PID %d) created

a child process with PID %d.\n", getpid(),

child_pid); Int status; Wait(&status);

If (WIFEXITED(status)) {

Printf("Child process exited with status %d.

\n", WEXITSTATUS(status)); } } Return 0;}

FCFS

#include<stdio.h>

Int main() {

```
Int bt[20], wt[20], tat[20], I,n;
Float wtavg, tatavg;
Printf("\nEnter the number of processes –");
Scanf("%d", &n); For(i=0;i<n;i++) {
printf("\nEnter Burst Time for Process %d –", i);
Scanf("%d", &bt[i]); } Wt[0] = wtavg = 0;
Tat[0] = tatavg = bt[0]; For(i=1;i<n;i++) {
Wt[i] = wt[i-1]+bt[i-1]; Tat[i] = tat[i-1] +bt[i];
Wtavg = wtavg + wt[i]; Tatavg = tatavg +tat[i]; }
Printf("\t PROCESS \tBURST TIME \t WAITING
TIME\t TURNAROUND TIME\n"); For(i=0;i<n;i++)
Printf("\n\t P%d \t\t %d \t\t %d \t\t %d", I,
bt[i], wt[i], tat[i]);
Prtf("\nAverage Waiting Time --%f", wtavg/n);
Prtf("\nAverage Turnaround Time --%f", tatavg/n);
Return 0;
```

SJF

```
#include<stdio.h> Int main() {
Int p[20], bt[20], wt[20], tat[20], I, k, n, temp;
Float wtavg, tatavg;
Printf("\nEnter the number of processes –");
Scanf("%d", &n) For(i=0;i<n;i++) { P[i]=I;
Printf("Enter Burst Time for Process %d –", i);
Scanf("%d", &bt[i]); }
For(i=0;i<n;i++) For(k=i+1;k<n;k++)
```

```
If(bt[i]>bt[k]) { Temp=bt[i]; Bt[i]=bt[k];
Bt[k]=temp; } Wt[0] = wtavg = 0;
Tat[0] = tatavg = bt[0]; For(i=1;i<n;i++) {
Temp=p[i];P[i]=p[k]; P[k]=temp;
Wt[i] = wt[i-1]+bt[i-1];
Tat[i] = tat[i-1] +bt[i]; Wtavg = wtavg + wt[i];
Tatavg = tatavg +tat[i]; }
Printf("\n\t PROCESS \tBURST TIME \t WAITING
TIME\t TURNAROUND TIME\n"); For(i=0;i<n;i++)
Pf("\n\t P%d \t\t %d \t\t %d \t\t %d",
p[i], bt[i], wt[i], tat[i]);
Printf("\nAverage Waiting Time --%f", wtavg/n);
Printf("\nAverage Turnaround Time --%f", tatavg/n);
Return 0;}
```

RR

```
#include<stdio.h> Int main() {
Int I,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
Float awt=0,att=0,temp=0;
Printf("Enter the no of processes –");
Scanf("%d",&n); For(i=0;i<n;i++) {
Printf("\nEnter Burst Time for process %d –", i+1);
Scanf("%d",&bu[i]); Ct[i]=bu[i];
Printf("\nEnter the size of time slice –");
Scanf("%d",&t); Max=bu[0]; For(i=1;i<n;i++)
If(max<bu[i]) Max=bu[i]; For(j=0;j<(max/t)+1;j++)
```

```
For(i=0;i<n;i++) If(bu[i]!=0) If(bu[i]<=t) {
Tat[i]=temp+bu[i]; Temp=temp+bu[i]; Bu[i]=0;}
Else { Bu[i]=bu[i]-t; Temp=temp+t;}
For(i=0;i<n;i++) { Wa[i]=tat[i]-ct[i];
Att+=tat[i]; Awt+=wa[i];}
Printf("\nThe Average Turnaround time is --%f",att/n);
Printf("\nThe Average Waiting time is --%f ",awt/n);
Printf("\n\tPROCESS\t BURST TIME \t WAITING TIME
\tTURNAROUND TIME\n"); For(i=0;i<n;i++)
Printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
Return 0; }
```

PRIORITY

```
#include<stdio.h> Int main() {
Int p[20],bt[20],pri[20], wt[20],tat[20],I, k, n, temp;
Float wtavg, tatavg;
Printf("Enter the number of processes ---");
Scanf("%d",&n); For(i=0;i<n;i++) { P[i] = I;
Printf("Enter the Burst Time & Priority of Process %d ---",i);
Scanf("%d %d",&bt[i], &pri[i]); } For(i=0;i<n;i++)
For(k=i+1;k<n;k++) If(pri[i] > pri[k]) { Temp=p[i];
P[i]=p[k]; P[k]=temp; Temp=bt[i]; Bt[i]=bt[k];
Bt[k]=temp; Temp=pri[i]; Pri[i]=pri[k]; Pri[k]=temp; }
Wtavg = wt[0] = 0; Tatavg = tat[0] = bt[0]; For(i=1;i<n;i++) {
Wt[i] = wt[i-1] + bt[i-1]; Tat[i] = tat[i-1] + bt[i];
Wtavg = wtavg + wt[i]; Tatavg = tatavg + tat[i];}
```

```
Printf(“\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING
TIME\tTURNAROUNDTIME”); For(i=0;i<n;i++)
Printf(“\n%d \t\t %d \t\t %d \t\t %d \t\t %d “,
p[i],pri[i],bt[i],wt[i],tat[i]);
Printf(“\nAverage Waiting Time is ---%f”,wtavg/n);
Printf(“\nAverage Turnaround Time is ---%f”,tatavg/n);
Return 0; frag[i]=temp; bf[ff[i]]=1; }
printf("\nFile_no:\tFile_size :\tBlock_no:\t
Block_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]);
```

PROGRAM:-03

```
#include <stdio.h> #include <stdlib.h> Int mutex = 1;
Int full = 0; Int empty = 10, data = 0; Void producer() {
--mutex; frag[i]=temp; bf[ff[i]]=1; }
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\t
Fragement"); 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]);++full; --empty; Data++;
Printf(“\nProducer produces item number: %d\n”, data);
++mutex; Void consumer() { --mutex; --full; ++empty;
Printf(“\nConsumer consumes item number: %d.\n”, data);
Data--; ++mutex;} Int main() { Int n, I;
Printf(“\n1. Enter 1 for Producer”“\n2. Enter 2 for Consumer”
“\n3. Enter 3 to Exit”); For (I = 1; I > 0; i++) {
```

```
Printf(“\nEnter your choice: “); Scanf(“%d”, &n);
Switch (n) { Case 1: if ((mutex == 1) && (empty != 0)) {
Producer(); frag[i]=temp; bf[ff[i]]=1; }
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\t
Fragement"); 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]);} Else {
Printf(“The Buffer is full. New data cannot be produced!”)}
Break;
Case 2:If ((mutex == 1) && (full != 0)) { Consumer(); } Else
Printf(“The Buffer is empty! New data cannot be consumed!”);
}Break; Case 3: Exit(0); Break;} } }
```

PROGRAM:-05

```
#include<stdio.h> #include<conio.h>
void main() {
int work[5],avl[5],alloc[10][10],l;
int need[10][10],n,m,l,j,avail[10],max[10][10],k,count,i,
fcount=0,pr[10]; char finish[10]={'f','f','f','f','f','f','f','f','f','f'};
printf("\n enter the no of process"); scanf("%d",&n);
printf("\n enter the no of resources"); scanf("%d",&m);
printf("\n enter the total no of resources");
for(i=1;i<=m;i++) scanf("%d",&avail[i]);
printf("\n enter the max resources req by each pr alloc matrix");
for(i=1;i<=n;i++) for(j=1;j<=m;j++) scanf("%d",&max[i][j]);
printf("\n process allocation matrix"); for(i=1;i<=n;i++)
```

```
for(j=1;j<=m;j++) scanf("%d",&alloc[i][j]);
for(i=1;i<=n;i++) for(j=1;j<=m;j++)
need[i][j]=max[i][j]-alloc[i][j]; for(i=1;i<=n;i++) { k=0;
for(j=1;j<=m;j++) { k=k+alloc[i][j]; } avl[i]=avl[i]-k;
work[i]=avl[i]; } for(k=1;k<=n;k++) for(i=1;i<=n;i++) {
count=0; for(j=1;j<=m;j++) {
if((finish[i]=='f')&&(need[i][j]<=work[i])) count++; }
if(count==m) { for(l=1;l<=m;l++)
work[l]=work[l]+alloc[i][l]; finish[i]='t'; pr[k]=l;
break; } } For(i=1;i<=n;i++) If(finish[i]=='t')
Fcount++; If(fcount==n){} Else
Printf("\n the system is in safe state");
For(i=1;i<=n;i++) Printf("\n %d",pr[i]);
Printf("\n the system is not in safe state");}
```

PROGRAM :BEST-FIT

```
#include<stdio.h> #define max 25
Void main() { Int frag[max],b[max],
f[max],l,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\t
BlockSize\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]); }
```

PROGRAM :FIRST-FIT

```
#include<stdio.h> #define max 25
Void main() { Int Frag[max],b[max],
f[max],I,j,nb,nf,temp,highes
T=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) { Temp=b[j]-f[i];
If(temp>=0) If(highest<temp){}}
Frag[i]=highest; bf[ff[i]]=1; highest=0;}
Ff[i]=j;highest=temp;}
Printf(“\nFile_no:\tFile_size:\tBlock_no:\t
Block_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]); }
```

PROGRAM:-WORST-FIT

```
#include<stdio.h> #define max 25
frag[i]=temp; bf[ff[i]]=1; }
printf("\nFile_no:\tFile_size :\tBlock_no:\t
Block_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]); Void main() {
Int Frag[max],b[max],f[max],I,j,nb,nf,t
Emp; 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:\t
Block_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]);
```

PROGRAM:-04

P1: WRITE FIRST

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
```

```
#include <unistd.h>
int main()
{
int fd;
char * myfifo = "/tmp/myfifo";
mkfifo(myfifo, 0666);
char arr1[80], arr2[80];
while (1)
{
fd = open(myfifo, O_WRONLY);
fgets(arr2, 80, stdin);
write(fd, arr2, strlen(arr2)+1);
close(fd);
fd = open(myfifo, O_RDONLY);
read(fd, arr1, sizeof(arr1));
printf("User2: %s\n", arr1);
close(fd);
return 0;
}
```

P2: READ FIRST

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
```

```
#include <unistd.h>
int main()
{
int fd1;
char * myfifo = "/tmp/myfifo";
mkfifo(myfifo, 0666);
charstr1[80], str2[80];
while (1)
{
fd1 = open(myfifo,O_RDONLY);
read(fd1, str1, 80);
printf("User1: %s\n", str1);
close(fd1);
fd1 = open(myfifo,O_WRONLY);
fgets(str2, 80, stdin);
write(fd1, str2, strlen(str2)+1);
close(fd1);
return 0;
}
```