```
Pratical no.2
                                                                                   break:
                                                                                                                             Set B
                                         tok[i]=NULL;
                                                                                                                            Write a program to implement a
Set A
Write a C program that behaves
                                                                                                                             toy shell (Command Interpreter).
like a shell which displays the
                                         void count(char *fn, char op)
                                                                                                                            It
                                                                                   int main()
command prompt 'myshell$'. It
                                                                                                                            has its own prompt say "MyShell
accepts the command, tokenize the
                                                                                   char buff[80],*args[10];
                                                                                                                            $". Any normal shell command is
                                         int fh,cc=0,wc=0,lc=0;
command line and execute it by
                                          char c:
                                                                                                                             executed
                                                                                   int pid;
                                         fh = open(fn,O RDONLY);
                                                                                                                            from this shell (MyShell$) by
creating the child process. Also
                                                                                   while(1)
implement the additional
                                          if(fh==-1)
                                                                                                                             starting a child process to execute
command 'count' as
                                                                                   printf("myshell$ ");
                                                                                                                            the system
1)myshell$ count c filename: It will
                                         printf("File %s not found.\n",fn);
                                                                                   fflush(stdin);
                                                                                                                            program corresponding to the
                                                                                   fgets(buff.80.stdin):
                                                                                                                            command. It should additionally
display the number of characters
                                         return:
                                                                                   buff[strlen(buff)-1]='\0';
in given file
                                                                                                                            interpret the
2)myshell$ count w filename: It
                                                                                   make toks(buff,args);
                                                                                                                            following commands:
                                         while(read(fh,&c,1)>0)
                                                                                   if(strcmp(args[0],"count")==0)
                                                                                                                            1) list f dirname: To print names of
will display the number of words
in given file
                                         if(c==' ') wc++;
                                                                                   count(args[2],args[1][0]);
                                                                                                                             all the files in current directory
3)myshell$ count l filename: It will
                                          else if(c == ' n')
                                                                                                                            2) list n dirname: To print the
                                                                                   else
display the number of lines in
                                                                                                                            number of all entries in the current
given file
                                                                                   pid = fork();
                                          wc++;
                                                                                                                            directory.
#include <sys/types.h>
                                          lc++;
                                                                                   if(pid>0)
                                                                                                                            3) list i dirname: To print names
#include <sys/stat.h>
                                                                                                                            and inodes of the files in the
                                                                                   wait():
#include <fcntl.h>
                                          cc++:
                                                                                   else
                                                                                                                            current directory.
#include <stdio.h>
                                                                                                                            #include <sys/types.h>
                                                                                                                            #include <sys/stat.h>
#include <stdlib.h>
                                          close(fh);
                                                                                   if(execvp(args[0],args)==-1)
#include <string.h>
                                                                                   printf("Bad command.\n");
                                                                                                                            #include <fcntl.h>
                                          switch(op)
void make toks(char *s, char *tok[])
                                                                                                                            #include <stdio.h>
                                          case 'c':
                                                                                                                            #include <stdlib.h>
                                                                                                                            #include <string.h>
int i=0;
                                          printf("No.of characters:%d\n",cc-1);
char *p;
                                         break:
                                                                                   return 0:
                                                                                                                            #include <dirent.h>
p = strtok(s, "");
                                          case 'w':
                                                                                                                             void make toks(char *s, char *tok[])
while(p!=NULL)
                                         printf("No.of words:%d\n",wc);
                                                                                                                            int i=0:
                                         break:
                                         case 'l':
tok[i++]=p;
                                                                                                                             char *p;
```

printf("No.of lines:%d\n",lc+1);

p=strtok(NULL," ");

```
#include <stdlib.h>
p = strtok(s, "");
                                                                                     else
                                                                                                                             #include <string.h>
while(p!=NULL)
                                           if(entry->d type==DT DIR) dc++;
                                           if(entry->d type==DT REG) fc++;
                                                                                     pid = fork();
                                                                                                                             #include <dirent.h>
 tok[i++]=p;
                                                                                                                             #include <unistd.h>
                                                                                     if(pid>0)
 p=strtok(NULL," ");
                                                                                      wait();
                                                                                                                             #include <fcntl.h> // Include this
                                          printf("%d Dir(s)\t%d
                                                                                                                             header for open(), O RDONLY, etc.
                                                                                     else
                                          File(s)\n'',dc,fc);
                                                                                      if(execvp(args[0],args)==-1)
                                          break;
                                                                                                                              int make toks(char *s, char *tok[]) {
tok[i]=NULL;
                                                                                      printf("Bad command.\n");
                                                                                                                               int i = 0:
                                          case 'i':
                                          while(entry=readdir(dp))
                                                                                                                               char *p;
void list(char *dn, char op)
                                                                                                                               p = strtok(s, "");
                                           if(entry->d type==DT REG)
                                                                                                                                while(p != NULL) {
                                            printf("%s\t%d\n",entry-
DIR *dp;
                                                                                   return 0:
                                                                                                                                  tok[i++] = p:
struct dirent *entry;
                                         >d name,entry->d fileno);
                                                                                                                                  p = strtok(NULL, " ");
int dc=0, fc=0;
                                                                                    SetC
                                                                                   1) Write a C program that behaves
                                                                                                                               tok[i] = NULL;
dp = opendir(dn);
                                                                                   like a shell which displays the
                                                                                                                                return i;
                                          closedir(dp);
if(dp==NULL)
                                                                                   command
                                                                                   prompt 'myshell$'. It
 printf("Dir %s not found.\n",dn);
                                                                                   accepts the command, tokenize the
                                                                                                                             void typeline(char *op, char *fn) {
                                                                                   command line and execute it by
                                                                                                                               int fh, i, j, n;
 return:
                                          int main()
                                                                                                                                char c:
                                                                                   creating
                                                                                                                               fh = open(fn, O RDONLY);
                                          char buff[80],*args[10];
                                                                                   the child process.
switch(op)
                                                                                                                               if(fh == -1) {
                                          int pid;
                                                                                    Also implement the additional
                                                                                   command 'typeline' as
                                                                                                                                  printf("File %s not found.\n",
                                                                                   1)myshell$ typeline n filename: It
case 'f':
                                                                                                                             fn);
                                          while(1)
 while(entry=readdir(dp))
                                                                                   will display first n lines of the file.
                                                                                                                                  return;
                                           printf("myshell$");
                                                                                   2)myshell$ typeline -n filename: It
 if(entry->d type==DT REG)
                                           fflush(stdin):
                                                                                   will display last n lines of the file.
  printf("%s\n",entry->d name);
                                                                                   3)myshell$ typeline a filename: It
                                           fgets(buff,80,stdin);
                                                                                                                                if(stremp(op, "a") == 0) {
                                                                                                                                  while(read(fh. &c. 1) > 0)
                                           buff[strlen(buff)-1]=\0;
                                                                                   will display all the lines of the file.
                                                                                                                                    printf("%c", c);
                                          make toks(buff,args);
                                                                                   #include <sys/types.h>
 break;
                                          if(strcmp(args[0],"list")==0)
case 'n':
                                                                                   #include <svs/stat.h>
                                                                                                                                  close(fh);
 while(entry=readdir(dp))
                                           list(args[2],args[1][0]);
                                                                                   #include <stdio.h>
                                                                                                                                  return;
```

```
printf("\nmyshell$");
                                                                                                 execlp(args[0], args[0],
                                               fgets(buff, 80, stdin);
                                                                                     args[1], args[2], args[3], NULL);
  n = atoi(op);
  if(n > 0) {
                                               buff[strlen(buff)-1] = '\0';
                                                                                               break:
                                               int n = make toks(buff, args);
    i = 0:
    while(read(fh, &c, 1) > 0) {
       printf("%c", c);
                                               switch (n) {
                                                                                                                                int i=0;
       if(c == '\n') i++;
                                                  case 1:
                                                                                       return 0;
                                                                                                                                char *p;
                                                    if(strcmp(args[0], "exit")
       if(i == n) break;
                                                                                     Q.2) Write a program to
                                          == 0)
                                                       exit(1);
                                                                                     implement a toy shell (Command
                                                    if (!fork())
                                                                                     Interpreter). It has
                                                       execlp(args[0], args[0],
                                                                                     its own prompt say "MyShell $".
  if(n < 0) {
                                          NULL);
    i = 0:
                                                                                     Any normal shell command is
    while(read(fh, &c, 1) > 0) {
                                                    break;
                                                                                     executed from
                                                                                     this shell (MyShell$) by starting a
       if(c == '\n') i++;
                                                  case 2:
                                                                                     child process to execute the system
    lseek(fh, 0, SEEK SET);
                                                    if (!fork())
                                                                                     program
                                                       execlp(args[0], args[0],
                                                                                     corresponding to the command. It
    i = 0;
    while(read(fh, &c, 1) > 0) {
                                          args[1], NULL);
                                                                                     should additionally interpret the
       if(c == '\n') j++;
                                                    break;
                                                                                     following
       if(i == i+n+1) break;
                                                                                     commands:
                                                  case 3:
                                                                                     1)search f file name :- To search
                                                    if (strcmp(args[0],
     while(read(fh, &c, 1) > 0) {
                                                                                     first occurrence of the pattern in
       printf("%c", c);
                                           "typeline") == 0)
                                                                                     the file
                                                                                     2)search a file name :- To search
                                                       typeline(args[1], args[2]);
                                                                                                                                return;
                                                    else -
                                                                                     all the occurrence of the pattern in
                                                       if (!fork())
                                                                                     the file
  close(fh);
                                                         execlp(args[0],
                                                                                     3)search c file name :- To count the
                                          args[0], args[1], args[2], NULL);
                                                                                     number of occurrence of the
                                                                                                                                case 'f':
                                                                                     pattern in the
int main() {
                                                    break:
                                                                                     file.
  char buff[80], *args[10];
                                                                                     #include <sys/types.h>
                                                                                     #include <sys/stat.h>
  while(1) {
                                                  case 4:
    printf("\n");
                                                    if (!fork())
                                                                                     #include <fcntl.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void make toks(char *s, char *tok[])
p = strtok(s, "");
while(p!=NULL)
tok[i++]=p;
p=strtok(NULL," ");
tok[i]=NULL;
void search(char *fn, char op, char
*pattern)
int fh,count=0,i=0,j=0;
char buff[255],c,*p;
fh = open(fn,O RDONLY);
if(fh==-1)
printf("File %s Not Found\n",fn);
switch(op)
 while(read(fh,&c,1))
 buff[i++]=c;
 if(c=='\n')
```

```
buff[i]='0';
                                             if(c=='\n')
  i=0;
  i++:
                                              buff[j]='0';
  if(strstr(buff,pattern))
                                              i = 0;
                                              i++;
   printf("%d: %s",i,buff);
                                              if(strstr(buff,pattern))
                                              printf("%d: %s",i,buff);
   break;
                                            }//switch
 break:
                                            close(fh);
case 'c':
                                            }//search
 while(read(fh,&c,1))
                                            int main()
 buff[i++]=c;
 if(c=='\n')
                                            char buff[80],*args[10];
                                            int pid;
  buff[i]='0';
  i=0;
                                            while(1)
  p = buff;
                                            printf("myshell$");
  while(p=strstr(p,pattern))
                                            fflush(stdin);
                                             fgets(buff,80,stdin);
   count++;
                                            buff[strlen(buff)-1]='\0';
   p++:
                                            make toks(buff,args);
                                            if(strcmp(args[0],"search")==0)
                                             search(args[3],args[1][0],args[2]);
 printf("Total No. of Occurrences
                                             else
= %d\n".count):
                                             pid = fork():
 break:
case 'a':
                                             if(pid>0)
 while(read(fh,&c,1))
                                              wait();
                                             else
 buff[i++]=c;
```

```
if(execvp(args[0],args)==-1)
  printf("Bad command.\n");
return 0;
Assignment 3- CPU Scheduling
SET A
1) Write the program to simulate
FCFS CPU-scheduling. The arrival
time and first CPU-burst for
different n number of processes
should
be input to the algorithm. Assume
that the fixed IO waiting time (2
units). The next CPU-burst should
be generated randomly. The
output should give Gantt chart,
turnaround time and waiting time
for each process. Also find the
average waiting time and
turnaround
time.
#include<stdio.h>
void findWaitingTime(int
processes[], int n, int bt[], int wt[]) {
    wt[0] = 0;
  // Calculating waiting time for
each process
  for (int i = 1; i < n; i++) {
```

```
wt[i] = bt[i-1] + wt[i-1];
void findTurnAroundTime(int
processes[], int n, int bt[], int wt[], int
tat[]) {
   for (int i = 0; i < n; i++) {
     tat[i] = bt[i] + wt[i];
void findavgTime(int processes[], int
n, int bt[]) 
  int wt[n], tat[n], total wt = 0,
total tat = 0;
     findWaitingTime(processes, n,
bt, wt);
   findTurnAroundTime(processes,
n, bt, wt, tat);
  printf("Processes Burst time
Waiting time Turn around time\n");
  for (int i = 0; i < n; i++) {
     total wt = total wt + wt[i];
     total tat = total tat + tat[i];
     printf(" %d\t\t%d\t\t%d\t\t%d\n",
(i+1), bt[i], wt[i], tat[i]);
```

```
float avg wt = (float)total wt /
(float)n:
  float avg tat = (float)total tat /
(float)n;
  printf("Average waiting time
= \%.2 \text{ f/n''}, \text{ avg wt)};
  printf("Average turn around time
= \%.2f\n'', avg tat);
int main() {
  int processes [] = \{1, 2, 3\};
  int n = \text{size} of processes / size of
processes[0];
   int burst time[] = \{10, 5, 8\};
  findavgTime(processes, n,
burst time);
  return 0;
2) Write a simulation program to
implement a Non-Pre-emptive
Shortest Job
First (SJF) – CPU scheduling
algorithm. Accept the number of
Processes
and arrival time and CPU burst
time for each process as input. The
output
```

```
should give the Gantt chart,
turnaround time and waiting time
for each
process. Also display the average
turnaround time and average
waiting time.
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct process info
char pname[20];
int at.bt.ct.bt1:
struct process info *next;
}NODE;
int n;
NODE *first, *last;
void accept info()
NODE *p;
int i:
printf("Enter no.of process:");
scanf("%d",&n);
for(i=0;i< n;i++)
p = (NODE^*)malloc(sizeof(NODE));
printf("Enter process name:");
scanf("%s",p->pname);
printf("Enter arrival time:");
scanf("\%d",\&p->at);
printf("Enter first CPU burst time:");
\operatorname{scanf}(\text{"%d"}, \&p->bt);
p->bt1 = p->bt;
p->next = NULL;
```

```
if(first==NULL)
                                            printf("%s\t%d\t%d\n",
first=p;
                                            p->pname,p->at,p->bt1);
else
                                            p = p - next;
last->next=p;
last = p;
                                            void sort()
                                           NODE *p,*q;
void print output()
                                            int t:
NODE *p;
                                            char name[20];
float avg tat=0,avg wt=0;
                                            p = first;
printf("pname\tat\tbt\tct\ttat\twt\n");
                                            while(p->next!=NULL)
p = first:
while(p!=NULL)
                                            q=p->next;
                                            while(q!=NULL)
int tat = p->ct-p->at;
int wt = tat-p->bt;
                                            if(p->at > q->at)
avg tat+=tat;
avg wt+=wt;
                                            strcpy(name,p->pname);
printf("%s\t%d\t%d\t%d\t%d\t%d\n",
                                            strcpy(p->pname,q->pname);
p \rightarrow pname, p \rightarrow at, p \rightarrow bt, p \rightarrow ct, tat, wt);
                                            strcpy(q->pname,name);
p=p->next;
                                           t = p->at;
                                            p->at = q->at;
printf("Avg TAT=%f\tAvg
                                            q->at=t;
WT=\%f\n'',
                                            t = p - bt;
avg tat/n,avg wt/n);
                                            p->bt=q->bt;
                                            q - bt = t;
void print input()
                                            t = p->ct;
                                            p->ct = q->ct;
NODE *p:
                                            q->ct=t;
p = first:
                                           t = p - bt1:
printf("pname\tat\tbt\n");
                                            p->bt1 = q->bt1;
while(p!=NULL)
                                            q->bt1=t;
```

```
q=q->next;
                                         while(n1!=n)
                                                                                   if(strcmp(s[i].pname,s1[i].pname)==
                                                                                                                             Job First (SJF) – CPU scheduling
                                                                                                                             algorithm. Accept the number of
                                                                                   s1[i].end = s[i].end;
                                                                                                                             Processes as input. Also accept
p=p->next;
                                         p = get sif();
                                         if(p==NULL)
                                                                                                                             arrival time and CPU burst time
                                                                                   else
                                                                                                                             for each process as input. The
                                                                                   s1[++i] = s[i];
                                                                                                                            output should give the Gantt chart,
int time;
                                         time++;
NODE * get sjf()
                                         s[k].start = prev;
                                                                                   printf("%d",s1[0].start);
                                                                                                                             turnaround time and waiting time
                                         strcpy(s[k].pname,"*");
                                                                                                                            for each process. Also display the
                                                                                   for(i=0;i<=i;i++)
                                         s[k].end = time;
NODE *p,*min p=NULL;
                                                                                                                             average turnaround time and
int min=9999;
                                         prev = time;
                                                                                   m = (s1[i].end - s1[i].start);
                                                                                                                             average waiting time.
p = first;
                                         k++;
                                                                                   for(k=0;k\le m/2;k++)
                                                                                                                             #include<stdio.h>
                                                                                   printf("-");
                                                                                                                             #include<stdlib.h>
while(p!=NULL)
                                                                                                                            #include<string.h>
                                          else
                                                                                   printf("%s",s1[i].pname);
                                                                                                                            typedef struct process info
if(p->at<=time && p->bt1!=0 &&
                                                                                   for(k=0;k<(m+1)/2;k++)
                                         time+=p->bt1;
p > bt1 < min)
                                                                                   printf("-");
                                         s[k].start = prev;
                                                                                   printf("%d",s1[i].end);
                                                                                                                             char pname[20];
                                         strcpy(s[k].pname, p->pname);
min = p > bt1;
                                                                                                                             int at,bt,ct,bt1;
                                         s[k].end = time;
                                                                                                                             struct process info *next;
min p = p;
                                         prev = time;
                                                                                                                             }NODE;
                                                                                   int main()
                                         k++;
                                                                                                                             int n;
p=p->next;
                                                                                                                            NODE *first.*last:
                                         p->ct = time:
                                                                                   accept info();
                                                                                                                            void accept info()
return min p;
                                         p->bt1=0;
                                                                                   sort();
                                         n1++;
                                                                                   sifnp();
struct gantt chart
                                                                                   print output();
                                                                                                                            NODE *p;
                                         print input();
                                                                                   print gantt chart();
                                                                                                                             int i;
                                                                                                                             printf("Enter no.of process:");
int start;
                                          sort();
                                                                                   return 0;
                                                                                                                            scanf("%d",&n);
char pname[30];
                                                                                                                             for(i=0;i< n;i++)
int end;
}s[100],s1[100];
                                         void print gantt chart()
                                                                                                                             p = (NODE*)malloc(sizeof(NODE));
int k:
                                                                                                                            printf("Enter process name:");
void sifnp()
                                         int i,j,m;
                                                                                                                            scanf("%s",p->pname);
                                         s1[0] = s[0];
                                                                                   SetB
                                         for(i=1,j=0;i< k;i++)
int prev=0,n1=0;
                                                                                   1)Write a simulation program to
                                                                                                                             printf("Enter arrival time:");
NODE *p;
                                                                                                                            scanf("\%d",\&p->at);
                                                                                   implement a Pre-emptive Shortest
```

```
void sifp()
printf("Enter first CPU burst time:");
                                          p = first;
                                                                                    t = p - bt1:
                                          printf("pname\tat\tbt\n");
scanf("%d",&p->bt);
                                                                                    p->bt1 = q->bt1;
p->bt1 = p->bt;
                                          while(p!=NULL)
                                                                                                                               int prev=0,n1=0;
                                                                                    q - bt1 = t;
p->next = NULL;
                                                                                                                              NODE *p;
                                          printf("%s\t%d\t%d\n",
if(first==NULL)
                                                                                                                               while(n1!=n)
                                                                                    q=q->next;
first=p;
                                          p \rightarrow pname, p \rightarrow at, p \rightarrow bt1);
else
                                          p = p->next;
                                                                                    p=p->next;
                                                                                                                              p = get sif();
                                                                                                                              if(p==NULL)
last->next=p;
last = p;
                                          void sort()
                                                                                    int time;
                                                                                                                               time++;
                                                                                    NODE * get sif()
                                                                                                                               s[k].start = prev;
                                          NODE *p,*q;
                                                                                                                              strcpy(s[k].pname,"*");
void print output()
                                          int t:
                                                                                    NODE *p,*min p=NULL;
                                                                                                                               s[k].end = time;
NODE *p;
                                          char name[20];
                                                                                    int min=9999;
                                                                                                                               prev = time;
float avg tat=0,avg wt=0;
                                          p = first;
                                                                                    p = first;
                                                                                                                               k++;
printf("pname\tat\tbt\tct\ttat\twt\n");
                                          while(p->next!=NULL)
                                                                                    while(p!=NULL)
p = first;
                                                                                                                               else
while(p!=NULL)
                                                                                    if(p->at<=time && p->bt1!=0 &&
                                          q=p->next;
                                          while(q!=NULL)
                                                                                    p->bt1<min)
                                                                                                                               time++;
int tat = p->ct-p->at;
                                                                                                                               s[k].start = prev;
                                                                                                                              strcpy(s[k].pname, p->pname);
int wt = tat-p->bt;
                                          if(p->at > q->at)
                                                                                    min = p->bt1;
                                                                                                                              s[k].end = time;
avg tat+=tat;
                                                                                    min p = p;
                                                                                                                               prev = time;
avg wt+=wt;
                                          strcpy(name,p->pname);
                                          strcpy(p->pname,q->pname);
printf("%s\t%d\t%d\t%d\t%d\t%d\n",
                                                                                                                              k++;
                                                                                    p=p->next;
                                                                                                                              p->ct = time;
p->pname,p->at,p->bt,p->ct,tat,wt);
                                          strcpy(q->pname,name);
p=p->next;
                                          t = p->at;
                                                                                    return min p;
                                                                                                                              p->bt1--;
                                          p->at = q->at;
                                                                                                                              if(p->bt1==0)
printf("Avg TAT=%f\tAvg
                                                                                                                              n1++;
                                          q->at=t;
                                                                                    struct gantt chart
WT=\%f\n''.
                                          t = p - bt;
avg tat/n,avg wt/n);
                                          p->bt=q->bt;
                                                                                    int start;
                                                                                                                              print input();
                                                                                    char pname[30];
                                          q - bt = t;
                                                                                                                               sort();
                                                                                    int end;
void print input()
                                          t = p->ct;
                                                                                    }s[100],s1[100];
                                          p->ct = q->ct;
NODE *p;
                                                                                                                              void print gantt chart()
                                          q->ct=t;
                                                                                    int k;
```

```
int i,j,m;
s1[0] = s[0];
for(i=1, i=0; i < k; i++)
if(strcmp(s[i].pname,s1[i].pname)==
s1[i].end = s[i].end;
else
s1[++j] = s[i];
printf("%d",s1[0].start);
for(i=0;i<=i;i++)
m = (s1[i].end - s1[i].start);
for(k=0;k\le m/2;k++)
printf("-");
printf("%s",s1[i].pname);
for(k=0;k<(m+1)/2;k++)
printf("-");
printf("%d",s1[i].end);
int main()
accept info();
sort();
sifp();
print output();
print gantt chart();
return 0;
```

2) Write a simulation program to implement a Pre-emptive Shortest

```
Job First (SJF) – CPU scheduling
algorithm. Accept the number of
Processes as input. Also accept
arrival time and CPU burst time
for each process as input. The
output should give the Gantt chart,
turnaround time and waiting time
for each process. Also display the
average turnaround time and
average waiting time.
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct process info
char pname[20];
int at,bt,ct,bt1;
struct process info *next;
}NODE;
int n;
NODE *first,*last:
void accept info()
NODE *p;
int i;
printf("Enter no.of process:");
scanf("%d",&n);
for(i=0;i< n;i++)
p = (NODE*)malloc(sizeof(NODE));
printf("Enter process name:");
scanf("%s",p->pname);
printf("Enter arrival time:");
scanf("\%d",\&p->at);
```

```
printf("Enter first CPU burst time:");
                                          p = first:
                                         printf("pname\tat\tbt\n");
scanf("%d",&p->bt);
p->bt1 = p->bt;
                                         while(p!=NULL)
p->next = NULL;
if(first==NULL)
                                          printf("%s\t%d\t%d\n",
first=p;
                                         p->pname,p->at,p->bt1);
else
                                          p = p - next;
last->next=p;
last = p;
                                          void sort()
void print output()
                                         NODE *p,*q;
                                          int t:
NODE *p;
                                          char name[20];
float avg tat=0,avg wt=0;
                                          p = first;
printf("pname\tat\tbt\tct\ttat\twt\n");
                                         while(p->next!=NULL)
p = first;
while(p!=NULL)
                                          q=p->next;
                                          while(q!=NULL)
int tat = p->ct-p->at;
int wt = tat-p->bt;
                                          if(p->at > q->at)
avg tat+=tat;
avg wt+=wt;
                                          strcpy(name,p->pname);
                                         strcpy(p->pname,q->pname);
printf("%s\t%d\t%d\t%d\t%d\t%d\n",
p->pname,p->at,p->bt,p->ct,tat,wt);
                                          strcpy(q->pname,name);
p=p->next;
                                         t = p->at;
                                         p->at = q->at;
printf("Avg TAT=%f\tAvg
                                          q->at = t;
WT=\%f\n''.
                                         t = p - bt:
avg tat/n, avg wt/n);
                                         p->bt=q->bt;
                                          q - bt = t;
void print input()
                                         t = p->ct;
                                         p->ct = q->ct;
NODE *p;
                                          q->ct=t;
```

```
void sifp()
t = p - bt1:
                                                                                                                             2) Write a simulation program to
                                                                                   int i,j,m;
                                                                                                                             implement Round Robin CPU
p->bt1 = q->bt1;
                                          int prev=0,n1=0;
                                                                                   s1[0] = s[0];
                                                                                                                             scheduling
q->bt1 = t;
                                         NODE *p;
                                                                                   for(i=1,j=0;i< k;i++)
                                                                                                                             algorithm for the given time
                                                                                                                             quantum as input. Also accept the
                                          while(n1!=n)
q=q->next;
                                                                                   if(strcmp(s[i].pname,s1[i].pname)==
                                                                                                                             number of
                                                                                                                             processes and arrival time and
p=p->next;
                                         p = get sif();
                                         if(p==NULL)
                                                                                   s1[i].end = s[i].end;
                                                                                                                             CPU burst time for each process as
                                                                                   else
                                                                                                                             input. The
                                          time++;
                                                                                   s1[++j] = s[i];
                                                                                                                             output should give the Gant Chart,
int time;
NODE * get sif()
                                         s[k].start = prev;
                                                                                                                             turnaround time and waiting time
                                         strcpy(s[k].pname,"*");
                                                                                   printf("%d",s1[0].start);
                                                                                                                             for
NODE *p,*min p=NULL;
                                         s[k].end = time;
                                                                                   for(i=0;i<=j;i++)
                                                                                                                             each process. Also display the
int min=9999;
                                         prev = time;
                                                                                                                             average turnaround time and
p = first;
                                          k++;
                                                                                   m = (s1[i].end - s1[i].start);
                                                                                                                             average waiting
while(p!=NULL)
                                                                                   for(k=0;k\le m/2;k++)
                                                                                                                             time.
                                                                                   printf("-");
                                                                                                                             #include<stdio.h>
                                          else
                                                                                   printf("%s",s1[i].pname);
if(p->at<=time && p->bt1!=0 &&
                                                                                                                             #include<stdlib.h>
p->bt1<min)
                                         time++;
                                                                                   for(k=0;k<(m+1)/2;k++)
                                                                                                                             #include<string.h>
                                         s[k].start = prev;
                                                                                   printf("-");
                                                                                                                             typedef struct process info
                                         strcpy(s[k].pname, p->pname);
                                                                                   printf("%d",s1[i].end);
min = p > bt1;
                                         s[k].end = time;
min_p = p;
                                                                                                                             char pname[20];
                                         prev = time;
                                                                                                                             int at,bt,ct,bt1;
                                                                                   int main()
p=p->next;
                                         k++;
                                                                                                                             struct process info *next;
                                                                                                                             NODE:
                                         p->ct = time;
return min p;
                                                                                                                             int n,ts;
                                         p->bt1--;
                                                                                   accept info();
                                         if(p->bt1==0)
                                                                                   sort();
                                                                                                                             NODE *first, *last;
struct gantt chart
                                         n1++;
                                                                                                                             void accept info()
                                                                                   sifp();
                                                                                   print output();
                                                                                                                             NODE *p;
int start:
                                         print input();
                                                                                   print gantt chart();
char pname[30];
                                          sort();
                                                                                   return 0;
                                                                                                                             int i:
int end;
                                                                                                                             printf("Enter no.of process:");
}s[100],s1[100];
                                                                                                                             scanf("%d",&n);
                                         void print gantt chart()
                                                                                                                             for(i=0;i < n;i++)
int k;
                                                                                   Set C
```

{	}	p->at=q->at;	first = first->next;
p = (NODE*)malloc(sizeof(NODE));	printf("Avg TAT=%f\tAvg	q->at=t;	t->next=NULL;
<pre>printf("Enter process name:");</pre>	$WT=%f\n",$	t = p->bt;	return t;
scanf("%s",p->pname);	avg_tat/n,avg_wt/n);	p->bt=q->bt;	}
<pre>printf("Enter arrival time:");</pre>	}	q->bt=t;	<pre>void addq(NODE *t)</pre>
scanf("%d",&p->at);	<pre>void print input()</pre>	t = p - ct;	{
<pre>printf("Enter first CPU burst time:");</pre>	{	p-ct = q-ct;	last->next = t;
scanf("%d",&p->bt);	NODE *p;	q->ct=t;	last = t;
p->bt1 = p->bt;	p = first;	$t = p \rightarrow bt1;$	}
p->next = NULL;	<pre>printf("pname\tat\tbt\n");</pre>	p->bt1 = q->bt1;	struct gantt chart
if(first==NULL)	while(p!=NULL)	q > bt1 = t;	{
first=p;	{	}	int start;
else	printf("%s\t%d\t%d\n",	q=q->next;	char pname[30];
last->next=p;	p->pname,p->at,p->bt1);	}	int end;
last = p;	p = p->next;	p=p->next;	}s[100],s1[100];
}	}	}	int k;
printf("Enter time slice:");	}	}	void rr()
scanf("%d",&ts);	void sort()	int time;	{
}	{	int is arrived()	int prev=0,n1=0;
void print output()	NODE *p,*q;	{	NODE *p;
{	int t;	NODE *p;	while $(n1!=n)$
NODE *p;	char name[20];	p = first;	{
float avg tat=0,avg wt=0;	p = first;	while(p!=NULL)	if(!is arrived())
<pre>printf("pname\tat\tbt\tct\ttat\twt\n");</pre>	while(p->next!=NULL)	{	{
p = first;	{	if(p->at<=time && p->bt1!=0)	time++;
while(p!=NULL)	q=p->next;	return 1;	s[k].start = prev;
{	while(q!=NULL)	p=p->next;	strcpy(s[k].pname,"*");
int tat = p ->ct- p ->at;	{	}	s[k].end = time;
int $wt = tat-p->bt$;	if(p->at > q->at)	return 0;	k++;
avg tat+=tat;	{	}	prev=time;
avg wt+=wt;	strcpy(name,p->pname);	NODE * delq()	}
$printf("%s\t%d\t%d\t%d\t%d\t%d\n",$	strcpy(p->pname,q->pname);	{	else
p->pname,p->at,p->bt,p->ct,tat,wt);	strcpy(q->pname,name);	NODE *t;	{
p=p->next;	$t = p \rightarrow at;$	t = first;	p = first;
1 1 /	1 /	,	1 /

```
while(1)
                                           s1[0] = s[0];
                                           for(i=1,j=0;i < k;i++)
if(p->at<=time && p->bt1!=0)
                                           if(strcmp(s[i].pname,s1[i].pname)==
break:
p = delq();
                                           0)
                                           s1[j].end = s[i].end;
addq(p);
                                           else
p = first;
                                           s1[++i] = s[i];
if(p->bt1 \le ts)
                                           printf("%d",s1[0].start);
time+=p->bt1;
                                           for(i=0;i<=j;i++)
p - bt1 = 0;
                                           m = (s1[i].end - s1[i].start);
                                           for(k=0;k\le m/2;k++)
else
                                           printf("-");
time+=ts;
                                           printf("%s",s1[i].pname);
                                           for(k=0;k<(m+1)/2;k++)
p->bt1-=ts;
                                           printf("-");
                                           printf("%d",s1[i].end);
p->ct = time;
s[k].start = prev;
strcpy(s[k].pname,p->pname);
s[k].end = time;
                                           int main()
k++;
                                           accept info();
prev = time;
if(p->bt1==0) n1++;
                                           sort();
                                           rr();
p = delq();
addq(p);
                                           print output();
                                           print gantt chart();
print input();
                                           return 0;
void print gantt chart()
int i,j,m;
```

```
Assignment 4
1. Write the simulation program to
implement demand paging and
show the page
scheduling and total number of
page faults for the following given
page reference string.
Give input n as the number of
memory frames.
Reference String:
12,15,12,18,6,8,11,12,19,12,6,8,12,1
5.19.8
1) Implement FIFO
#include<stdio.h>
#define MAX 20
frames[MAX],ref[MAX],mem[MAX]
[MAX], faults, sp, m, n;
void accept()
int i:
printf("Enter no.of frames:");
scanf("%d", &n);
printf("Enter no.of references:");
scanf("%d", &m);
printf("Enter reference string:\n");
for(i=0;i < m;i++)
printf("[%d]=",i);
scanf("%d",&ref[i]);
void disp()
```

```
int i,j;
for(i=0;i \le m;i++)
printf("%3d",ref[i]);
printf("\n\n");
for(i=0;i< n;i++)
for(j=0;j < m;j++)
if(mem[i][j])
printf("%3d",mem[i][j]);
else
printf(" ");
printf("\n");
printf("Total Page
Faults: %d\n",faults);
int search(int pno)
int i:
for(i=0;i < n;i++)
if(frames[i]==pno)
return i;
return -1;
void fifo()
int i,j;
for(i=0;i < m;i++)
if(search(ref[i])==-1)
```

```
scanf("%d",&ref[i]);
frames[sp] = ref[i];
                                                                                     int i,min i,min=9999;
                                                                                                                                sp = get lru();
sp = (sp+1)\%n;
                                                                                     for(i=0;i< n;i++)
                                                                                                                                frames[sp] = ref[i];
                                          void disp()
                                                                                                                               time[sp] = i;
faults++;
for(j=0;j< n;j++)
                                                                                     if(time[i]<min)
                                                                                                                                faults++;
mem[j][i] = frames[j];
                                                                                                                                for(j=0;j< n;j++)
                                           int i,j;
                                          for(i=0;i<m;i++)
                                                                                     min = time[i];
                                                                                                                                mem[j][i] = frames[j];
                                          printf("%3d",ref[i]);
                                                                                     min i = i;
                                          printf("\n\n");
                                                                                                                                else
                                                                                                                                time[k]=i;
int main()
                                          for(i=0;i< n;i++)
                                                                                     return min i;
                                           for(j=0;j< m;j++)
accept();
                                                                                     void lru()
fifo();
                                                                                                                                int main()
                                          if(mem[i][j])
disp();
                                          printf("%3d",mem[i][j]);
return 0;
                                                                                     int i,j,k;
                                                                                                                                accept();
                                           else
                                                                                     for(i=0;i \le m \&\& sp \le n;i++)
                                                                                                                                lru();
2) Implement LRU
                                           printf(" ");
                                                                                                                                disp();
                                                                                     k=search(ref[i]);
#include<stdio.h>
                                                                                                                                return 0;
#define MAX 20
                                          printf("\n");
                                                                                     if(k==-1)
                                                                                                                                Set B
int
                                          printf("Total Page
frames[MAX],ref[MAX],mem[MAX]
                                                                                     frames[sp]=ref[i];
                                                                                                                                1. Write the simulation program to
                                          Faults: %d\n",faults);
[MAX], faults,
                                                                                     time[sp]=i;
                                                                                                                                implement demand paging and
sp,m,n,time[MAX];
                                                                                     faults++;
                                                                                                                                show the page
void accept()
                                           int search(int pno)
                                                                                                                                scheduling and total number of
                                                                                     sp++;
                                                                                     for(j=0;j< n;j++)
                                                                                                                                page faults for the following given
                                                                                     mem[j][i]=frames[j];
int i;
                                           int i;
                                                                                                                                page reference string.
printf("Enter no.of frames:");
                                           for(i=0;i< n;i++)
                                                                                                                                Give input n as the number of
scanf("%d", &n);
                                                                                     else
                                                                                                                                memory frames
printf("Enter no.of references:");
                                          if(frames[i]==pno)
                                                                                     time[k]=i;
                                                                                                                                Reference String:
scanf("%d", &m);
                                          return i:
                                                                                                                                12,15,12,18,6,8,11,12,19,12,6,8,12,1
                                                                                     for(;i<m;i++)
printf("Enter reference string:\n");
                                                                                                                                5,19,8
for(i=0;i < m;i++)
                                          return -1;
                                                                                                                                Implement OPT #include<stdio.h>
                                                                                     k = search(ref[i]);
                                                                                                                                int main()
printf("[%d]=",i);
                                          int get lru()
                                                                                     if(k==-1)
```

```
int no of frames, no of pages,
frames[10], pages[30], temp[10],
                                             if(frames[i] == -1)
                                                                                         if(flag3 == 0)
                                                                                                                                      frames[MAX],ref[MAX],mem[MAX]
flag1, flag2, flag3, i, j, k, pos, max,
                                                                                                                                      [MAX], faults,
                                                                                         max = temp[0];
                                                                                                                                      sp,m,n,count[MAX];
faults = 0:
                                             faults++;
printf("Enter number of frames: ");
                                             frames[i] = pages[i];
                                                                                         pos = 0;
  scanf("%d",&no of frames);
                                                                                         for(j = 1; j < no of frames; ++i)
                                             flag2 = 1;
                                                                                                                                      void accept()
                                             break;
printf("Enter number of pages: ");
                                                                                         if(temp[j] > max)
                                                                                                                                       int i;
scanf("%d", &no of pages);
                                                                                         max = temp[i];
                                                                                                                                       printf("Enter no.of frames:");
printf("Enter page reference string: ");
                                             if(flag2 == 0)
                                                                                         pos = j;
                                                                                                                                       scanf("%d", &n);
for (i = 0; i < no \text{ of pages}; ++i)
                                                                                                                                       printf("Enter no.of references:");
                                             flag3 = 0;
                                                                                                                                       scanf("%d", &m);
                                             for (j = 0; j < no \text{ of frames}; ++j)
scanf("%d", &pages[i]);
                                                                                         frames[pos] = pages[i];
                                            temp[j] = -1;
                                                                                         faults++;
                                                                                                                                       printf("Enter reference string:\n");
for (i = 0; i < no \text{ of frames}; ++i)
                                                                                                                                       for(i=0;i < m;i++)
                                             for(k = i + 1; k < no \text{ of pages}; ++k)
                                                                                         printf("\n");
frames[i] = -1;
                                                                                                                                       printf("[%d]=",i);
                                             if(frames[j] == pages[k])
                                                                                         for (i = 0; i < no \text{ of frames}; ++i)
                                                                                                                                       scanf("%d",&ref[i]);
for (i = 0; i < no \text{ of pages}; ++i)
                                            temp[j] = k;
                                                                                         printf("%d\t", frames[i]);
flag1 = flag2 = 0;
                                             break;
                                                                                                                                      void disp()
for(j = 0; j < no of frames; ++j)
                                                                                         printf("\nTotal Page Faults = %d",
                                                                                         faults);
                                                                                                                                       int i,j;
if(frames[i] == pages[i])
                                             for (j = 0; j < no \text{ of frames}; ++j)
                                                                                                                                       for(i=0;i < m;i++)
                                                                                         return 0;
flag1 = flag2 = 1; break;
                                             if(temp[i] == -1)
                                                                                                                                       printf("%3d",ref[i]);
                                                                                         MFU ANSWER
                                             pos = j;
                                                                                         #include<stdio.h>
                                                                                                                                       printf("\n\n");
if(flag1 == 0)
                                             flag3 = 1;
                                                                                         #define MAX 20
                                             break;
                                                                                                                                       for(i=0;i < n;i++)
for (j = 0; j < no \text{ of frames}; ++j)
```

```
for(j=0;j< m;j++)
                                            if(count[i]>max)
 if(mem[i][j])
                                             max = count[i];
  printf("%3d",mem[i][j]);
                                             \max i = i;
 else
  printf(" ");
                                            i=(i+1)\%n;
                                            }while(i!=sp);
 printf("\n");
                                            return max i;
printf("Total Page
Faults: %d\n",faults);
                                           void mfu()
                                           int i,j,k;
int search(int pno)
                                            for(i=0;i \le m \&\& sp \le n;i++)
int i;
                                            k=search(ref[i]);
for(i=0;i< n;i++)
                                            if(k==-1)
 if(frames[i]==pno)
                                             frames[sp]=ref[i];
                                             count[sp]++;
 return i;
                                             faults++;
                                             sp++;
return -1;
                                             for(j=0;j< n;j++)
                                             mem[j][i]=frames[j];
int get mfu(int sp)
                                            else
int i,max i,max=-9999;
                                             count[k]++;
i=sp;
do
                                            sp=0;
                                            for(;i \le m;i++)
```

```
k = search(ref[i]);
 if(k==-1)
 sp = get mfu(sp);
 frames[sp] = ref[i];
 count[sp]=1;
 faults++;
 sp = (sp+1)\%n;
 for(j=0;j< n;j++)
 mem[i][i] = frames[i];
 else
 count[k]++;
int main()
accept();
mfu();
disp();
return 0;
```

```
Practical Assignments:1
Set A
(1) Implement the C Program to
create a child process using fork(),
display parent and child process id.
Child process will display the
message "I am Child Process" and
the parent process should display
"I
am Parent Process".
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
// fork() Create a child process
int pid = fork();
if (pid > 0) {
printf("I am Parent process\n");
printf("ID : %d\n\n", getpid());
else if (pid == 0) {
printf("I am Child process\n");
// getpid() will return process id of
child process
printf("ID: %d\n", getpid());
else {
printf("Failed to create child
process");
return 0;
```

2)Write a program that demonstrates the use of nice() system call. After a child process is started using fork(), assign higher priority to the child using nice() system call.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
pid t pid:
pid = fork():
if (pid == 0)
printf("\nI am child process,
id=%d\n",getpid());
printf("\nPriority :%d,id=%d\n",nice
(-7),getpid());
else
printf("\nI am parent process,
id=%d\n",getpid());
nice(1);
printf("\nPriority :%d,id=%d\n",nice
(15),getpid());
return 0:
```

Set B:

1) Implement the C program to accept n integers to be sorted.

Main function creates child process using fork system call. Parent process sorts the integers using bubble sort and waits for child process using wait system call. Child process sorts the integers using insertion sort.

```
#include<stdio.h>
#include<sys/types.h>
#include<sunistd.h>
#include<stdlib.h>

void bubblesort(int arr[30],int n)
{
   int i,j,temp;
   for(i=0;i<n;i++)
   {
      for(j=0;j<n-1;j++)
      {
        if(arr[j]>arr[j+1])
      {
        temp=arr[j];
        arr[j]=arr[j+1];
        arr[j+1]=temp;
      }
   }
}
```

```
void insertionsort(int arr[30], int n)
  int i, j, temp;
  for (i = 1; i < n; i++)
     temp = arr[i];
     i = i - 1;
     while(j \ge 0 \&\& temp \le arr[j])
        arr[i+1] = arr[i];
       j = j-1;
     arr[j+1] = temp;
void fork1()
 int arr[25],arr1[25],n,i,status;
 printf("\nEnter the no of values in
array:");
 scanf("%d",&n);
 printf("\nEnter the array
elements:");
 for(i=0;i< n;i++)
   scanf("%d",&arr[i]);
 int pid=fork();
 if(pid==0)
     sleep(10);
printf("\nchild process\n");
printf("child process
id=%d\n",getpid());
insertionsort(arr,n);
```

```
printf("\nElements Sorted Using
insertionsort:"):
printf("\n");
for(i=0;i < n;i++)
printf("%d,",arr[i]);
printf("\b");
printf("\nparent process
id=%d\n",getppid());
system("ps -x");
else
printf("\nparent process\n");
printf("\nparent process
id=%d\n",getppid());
bubblesort(arr,n);
printf("Elements Sorted Using
bubblesort:");
    printf("\n");
     for(i=0;i< n;i++)
    printf("%d,",arr[i]);
   printf("\n\n");
int main()
fork1();
return 0;
```

```
2) Write a C program to illustrate
the concept of orphan process.
Parent
process creates a
child and terminates before child
has finished its task. So child
process
becomes orphan
process. (Use fork(), sleep(),
getpid(), getppid()).
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
int pid;
pid=getpid();
printf("Current Process ID
is : %d\n",pid);
printf("\n[Forking Child Process ... ]
\n");
pid=fork();
if(pid < 0)
printf("\nProcess can not be created
");
else
if(pid==0)
printf("\nChild Process is
Sleeping ...");
sleep(5);
```

```
printf("\nOrphan Child's Parent
ID : %d",getppid());
}
else
{ /* Parent Process */
printf("\nParent Process
Completed ...");
}
}
return 0;
}
```