PRACTICAL FILE

OPERATING SYSTEMS

BSc. (H) Computer Science
Third Semester

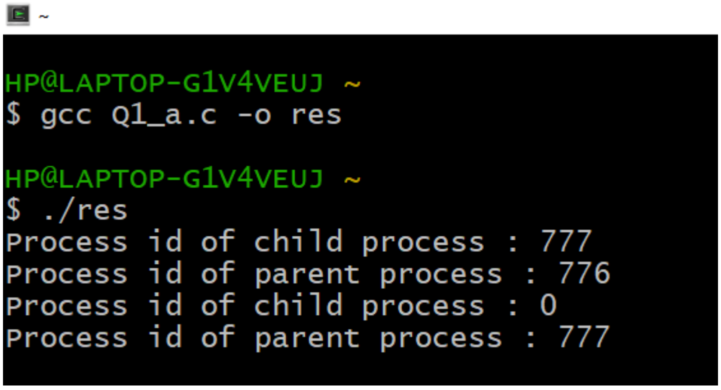
SUBMITTED BY:	SUBMITTED TO :
NAME : PAVANI BINDAL	DR. SHIKHA AGARWAL
ROLL NO. : 21/CS/31	

1. Write a program (using fork () and/or exec () commands) where parent and child execute:

a) same program, same code.

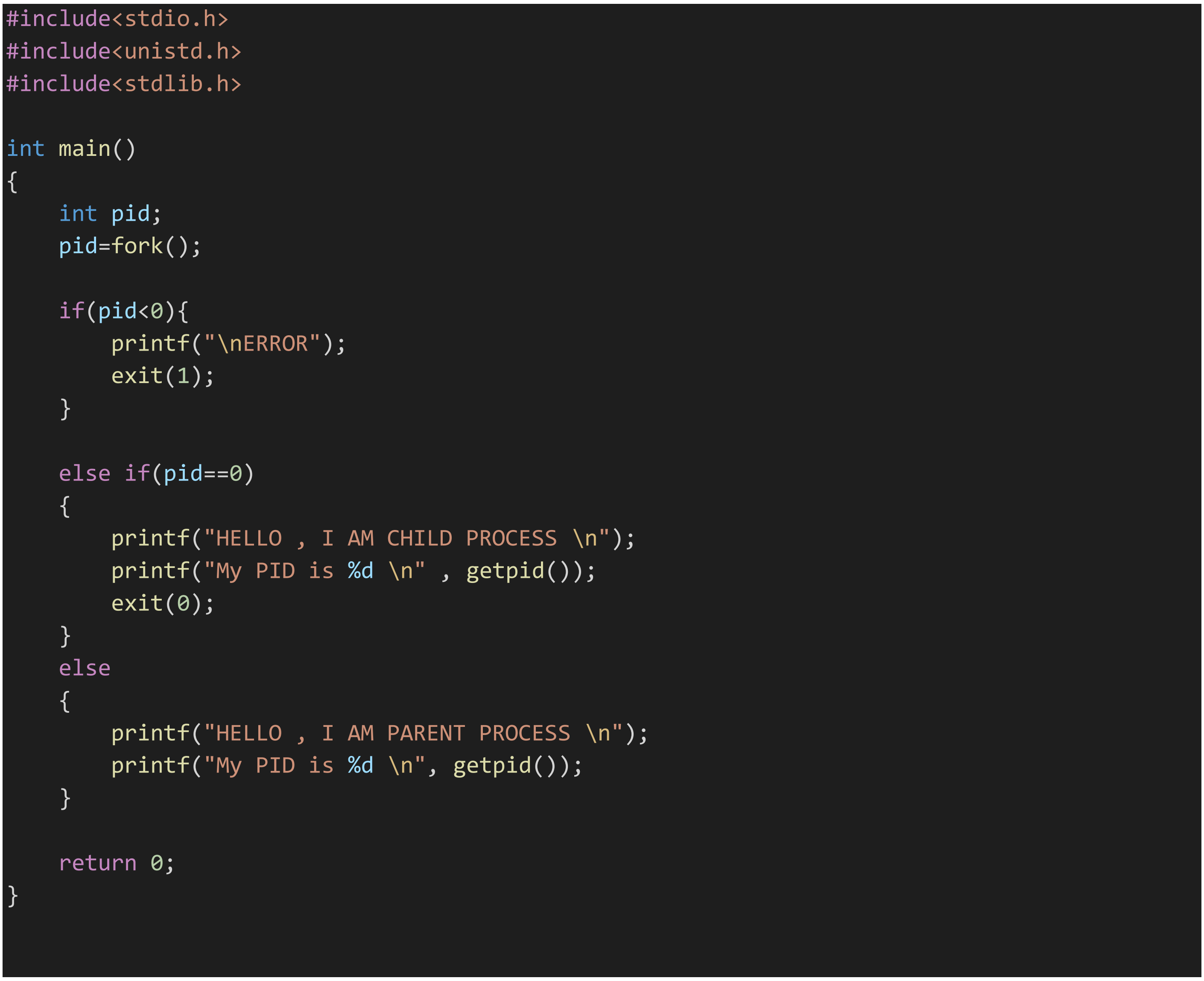
```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main()
{
    pid_t pid , p;
    p = fork();
    pid = getpid();

    if(p<0){
        fprintf(stderr ,"FORK FAILED");
        return 1;
    }

    printf("Process id of child process : %d \n" ,p);
    printf("Process id of parent process : %d \n" ,pid);
    return 0;
}
```

OUTPUT:



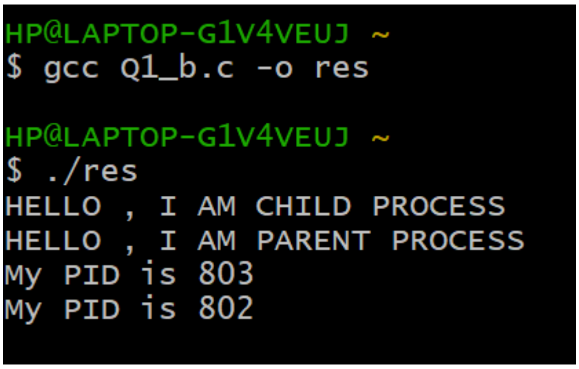
```
~
HP@LAPTOP-G1V4VEUJ ~
$ gcc Q1_a.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res
Process id of child process : 777
Process id of parent process : 776
Process id of child process : 0
Process id of parent process : 777
```

b) same program, different code.



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

int main()
{
    int pid;
    pid=fork();

    if(pid<0){
        printf("\nERROR");
        exit(1);
    }

    else if(pid==0)
    {
        printf("HELLO , I AM CHILD PROCESS \n");
        printf("My PID is %d \n" , getpid());
        exit(0);
    }
    else
    {
        printf("HELLO , I AM PARENT PROCESS \n");
        printf("My PID is %d \n", getpid());
    }

    return 0;
}
```

OUTPUT:

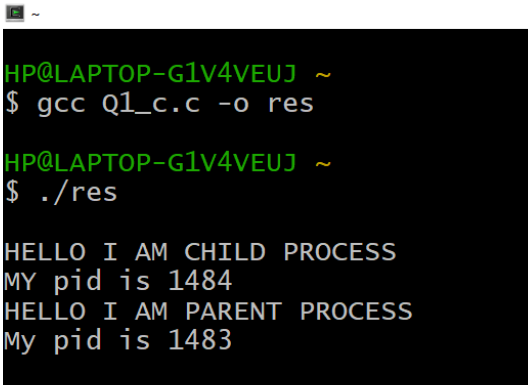


```
HP@LAPTOP-G1V4VEUJ ~
$ gcc Q1_b.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res
HELLO , I AM CHILD PROCESS
HELLO , I AM PARENT PROCESS
My PID is 803
My PID is 802
```

c) before terminating, the parent waits for the child to finish its task.

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

int main()
{
    int pid;
    pid=fork();
    if(pid<0)
    {
       printf("\nERROR");
       exit(1);
    }

    else if(pid==0)
    {
       printf("\nHELLO I AM CHILD PROCESS");
       printf("\nMY pid is %d" , getpid());
       exit(0);
    }
    else if(pid>0)
    {
       wait(NULL);
       printf("\nHELLO I AM PARENT PROCESS");
       printf("\nMy pid is %d \n" , getpid());
       exit(1);
    }
}
```

OUTPUT:

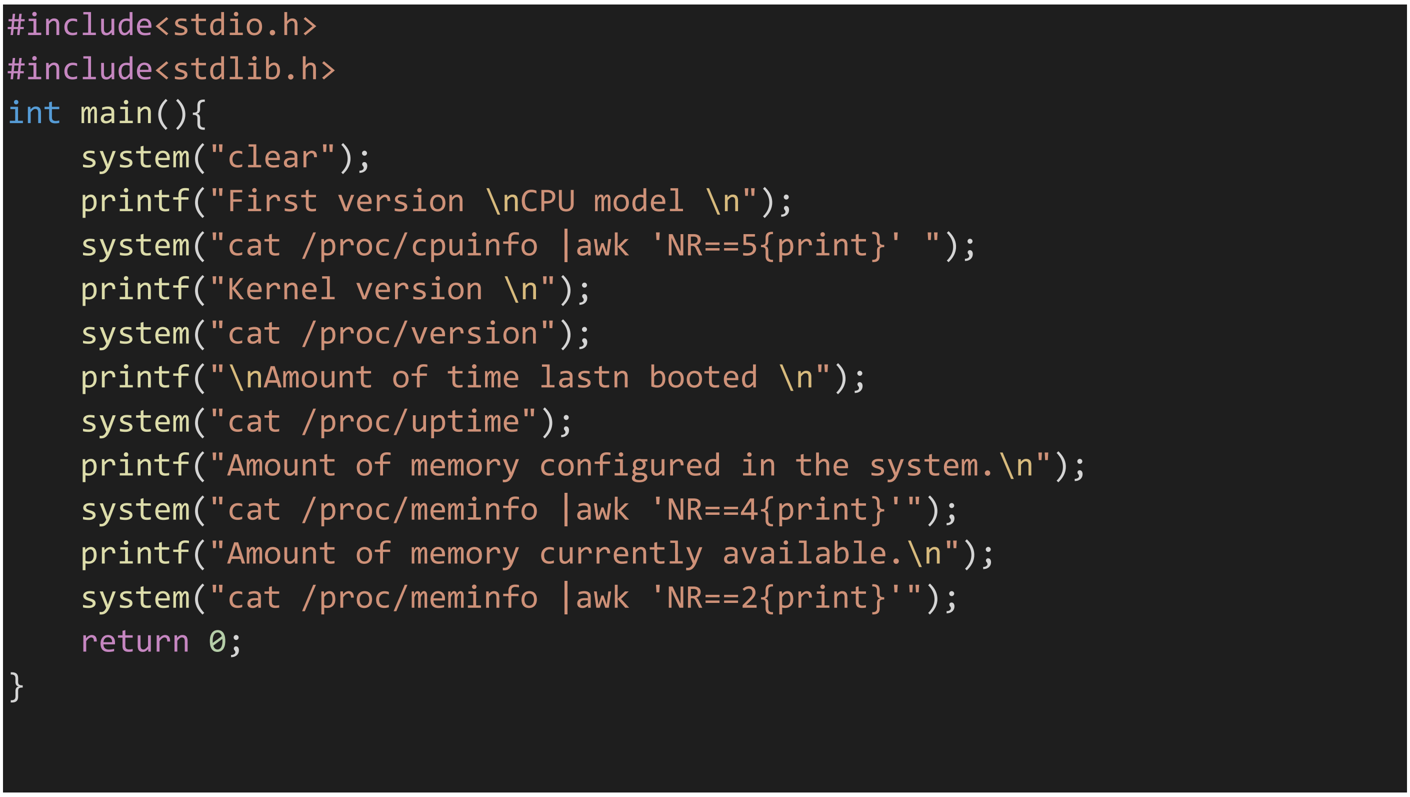


```
HP@LAPTOP-G1V4VEUJ ~
$ gcc Q1_c.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

HELLO I AM CHILD PROCESS
MY pid is 1484
HELLO I AM PARENT PROCESS
My pid is 1483
```

Q2. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)

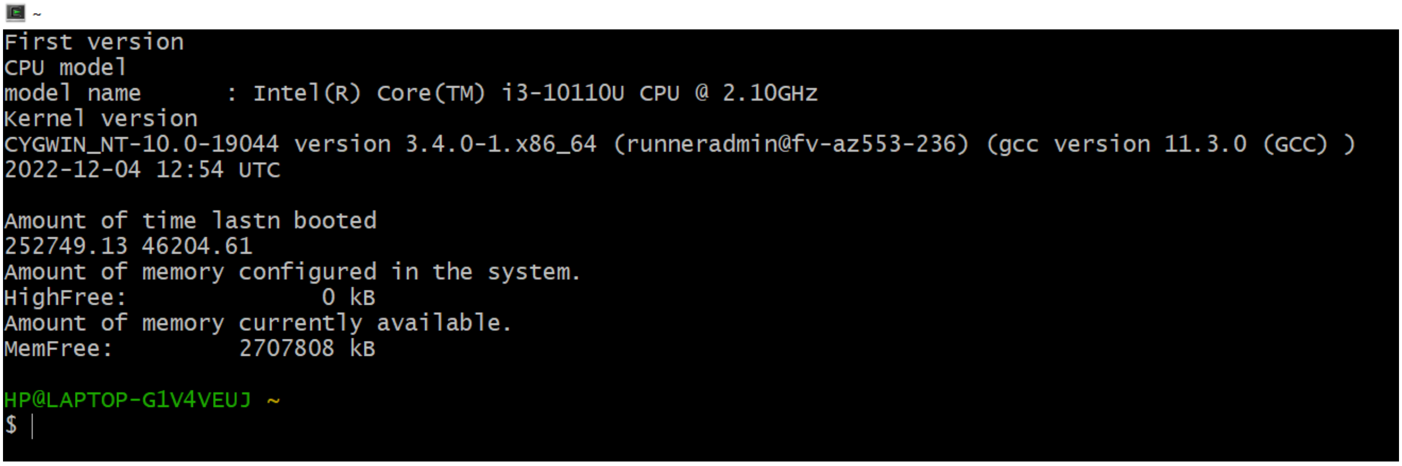
AND

Q3. Write a program to report behaviour of Linux kernel including information on 19 configured memory, amount of free and used memory. (Memory information)



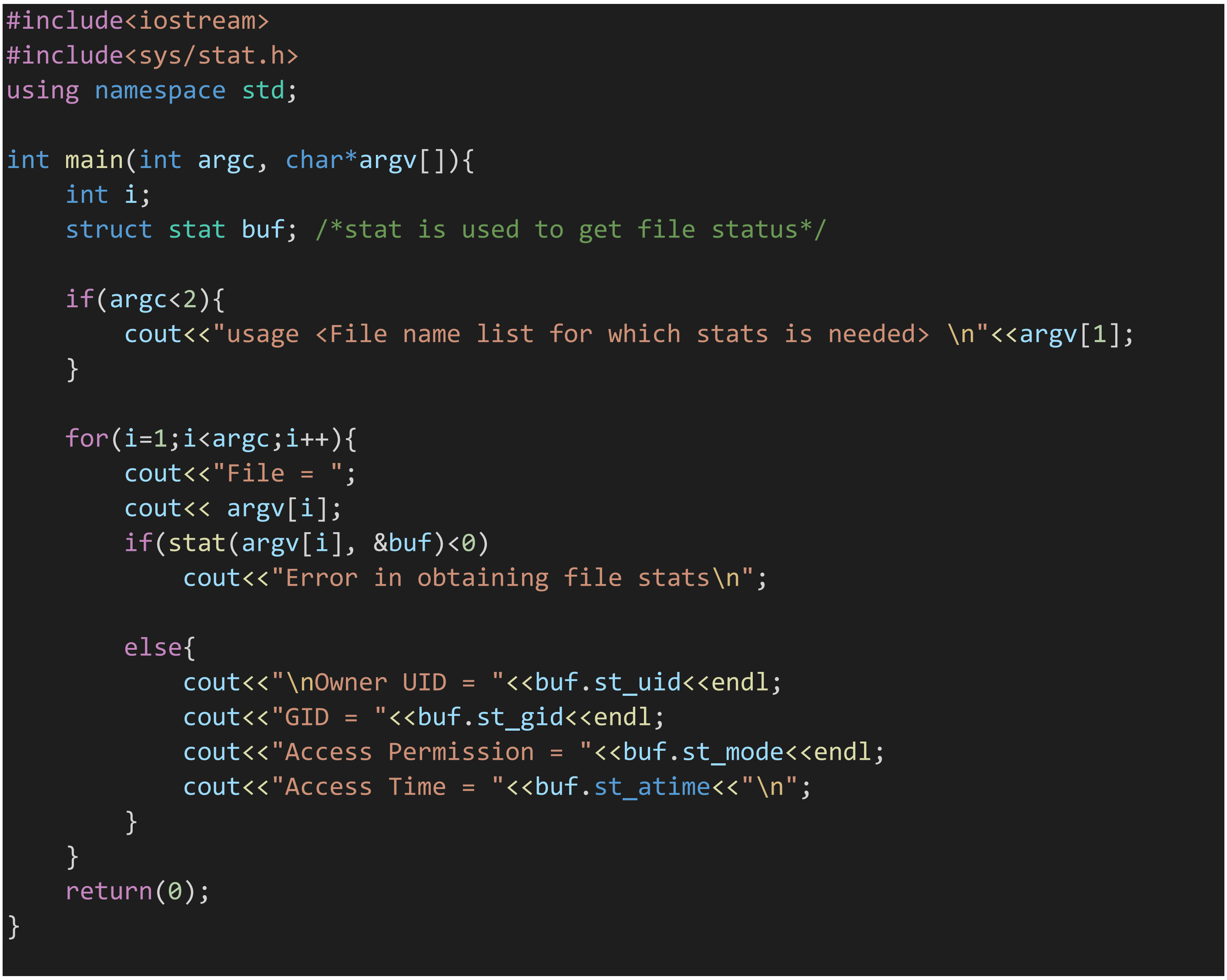
```c
#include<stdio.h>
#include<stdlib.h>
int main(){
    system("clear");
    printf("First version \nCPU model \n");
    system("cat /proc/cpuinfo |awk 'NR==5{print}' ");
    printf("Kernel version \n");
    system("cat /proc/version");
    printf("\nAmount of time lastn booted \n");
    system("cat /proc/uptime");
    printf("Amount of memory configured in the system.\n");
    system("cat /proc/meminfo |awk 'NR==4{print}'");
    printf("Amount of memory currently available.\n");
    system("cat /proc/meminfo |awk 'NR==2{print}'");
    return 0;
}
```

OUTPUT:



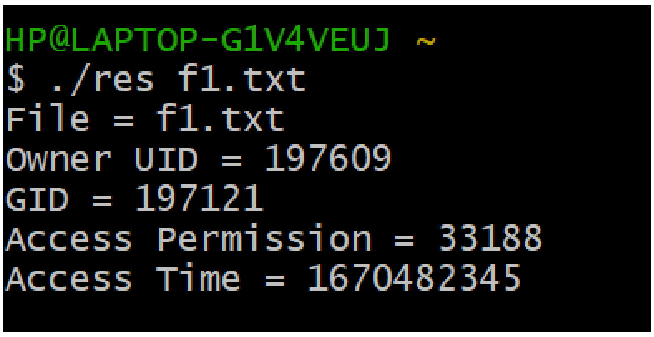
```
First version
CPU model
model name      : Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz
Kernel version
CYGWIN_NT-10.0-19044 version 3.4.0-1.x86_64 (runneradmin@fv-az553-236) (gcc version 11.3.0 (GCC) )
2022-12-04 12:54 UTC

Amount of time lastn booted
252749.13 46204.61
Amount of memory configured in the system.
HighFree:              0 kB
Amount of memory currently available.
MemFree:         2707808 kB

HP@LAPTOP-G1V4VEUJ ~
$ |
```

Q4. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.



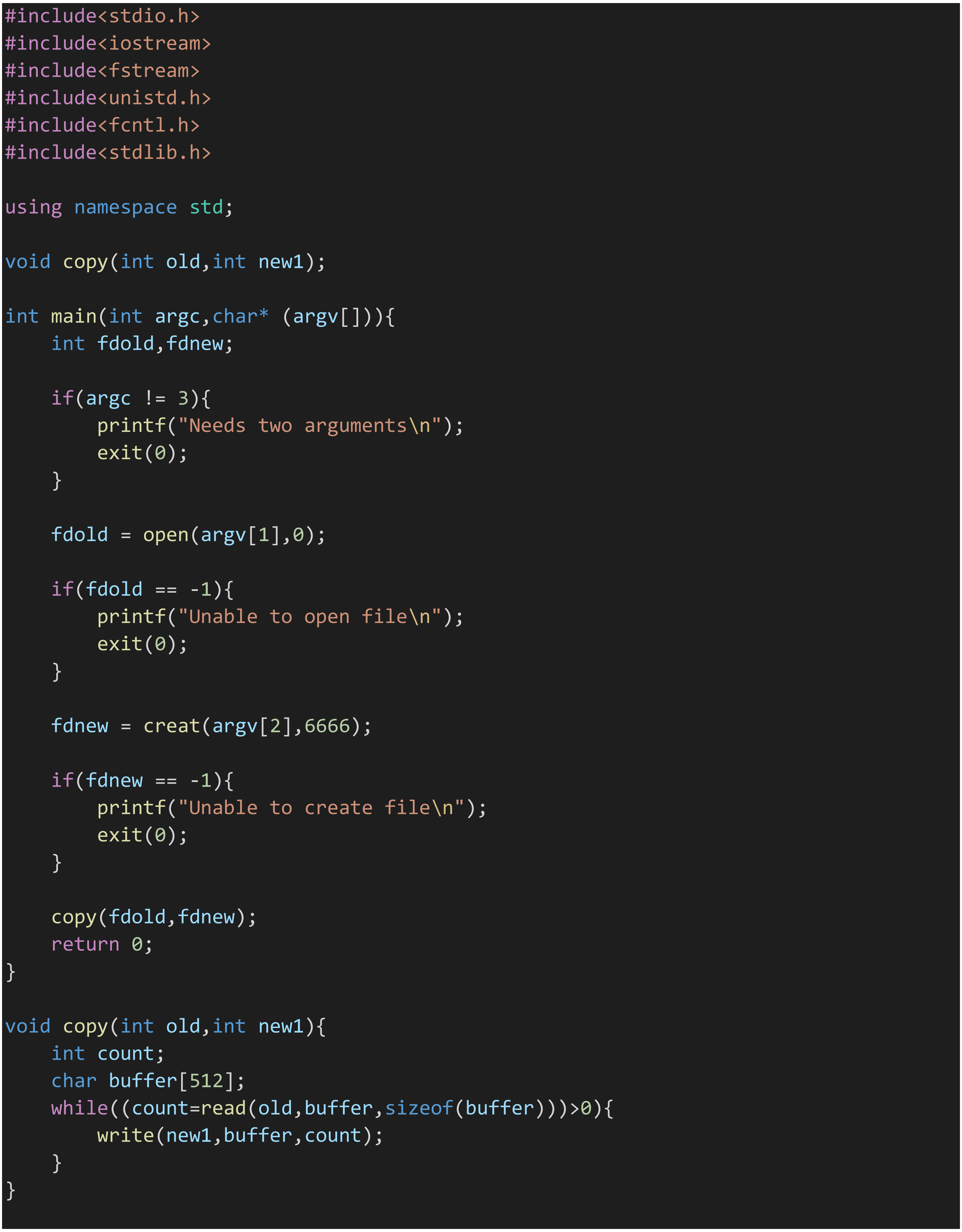
```
#include<iostream>
#include<sys/stat.h>
using namespace std;

int main(int argc, char*argv[]){
    int i;
    struct stat buf; /*stat is used to get file status*/

    if(argc<2){
        cout<<"usage <File name list for which stats is needed> \n"<<argv[1];
    }

    for(i=1;i<argc;i++){
        cout<<"File = ";
        cout<< argv[i];
        if(stat(argv[i], &buf)<0)
            cout<<"Error in obtaining file stats\n";

        else{
            cout<<"\nOwner UID = "<<buf.st_uid<<endl;
            cout<<"GID = "<<buf.st_gid<<endl;
            cout<<"Access Permission = "<<buf.st_mode<<endl;
            cout<<"Access Time = "<<buf.st_atime<<"\n";
        }
    }
    return(0);
}
```

OUTPUT:



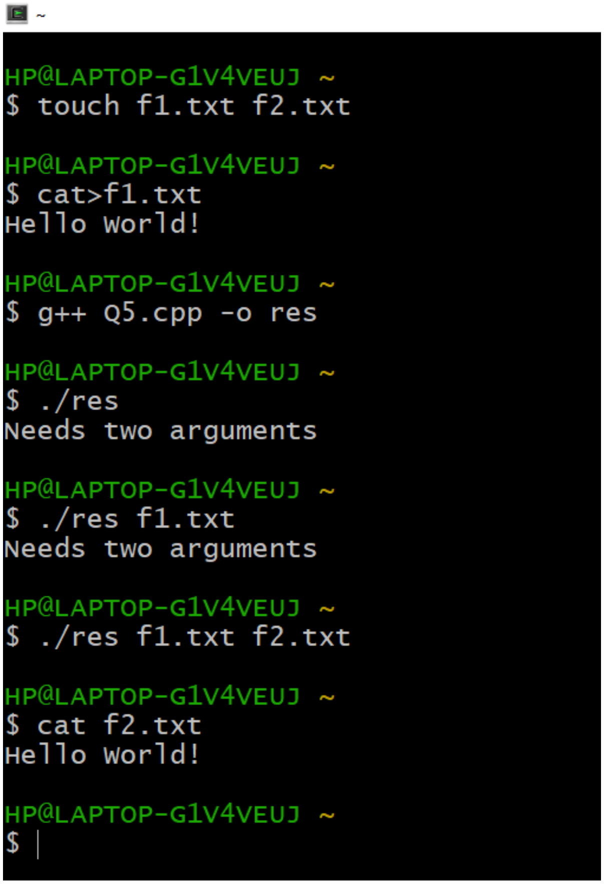
```
HP@LAPTOP-G1V4VEUJ ~
$ ./res f1.txt
File = f1.txt
Owner UID = 197609
GID = 197121
Access Permission = 33188
Access Time = 1670482345
```

Q5. Write a program to copy files using system calls.



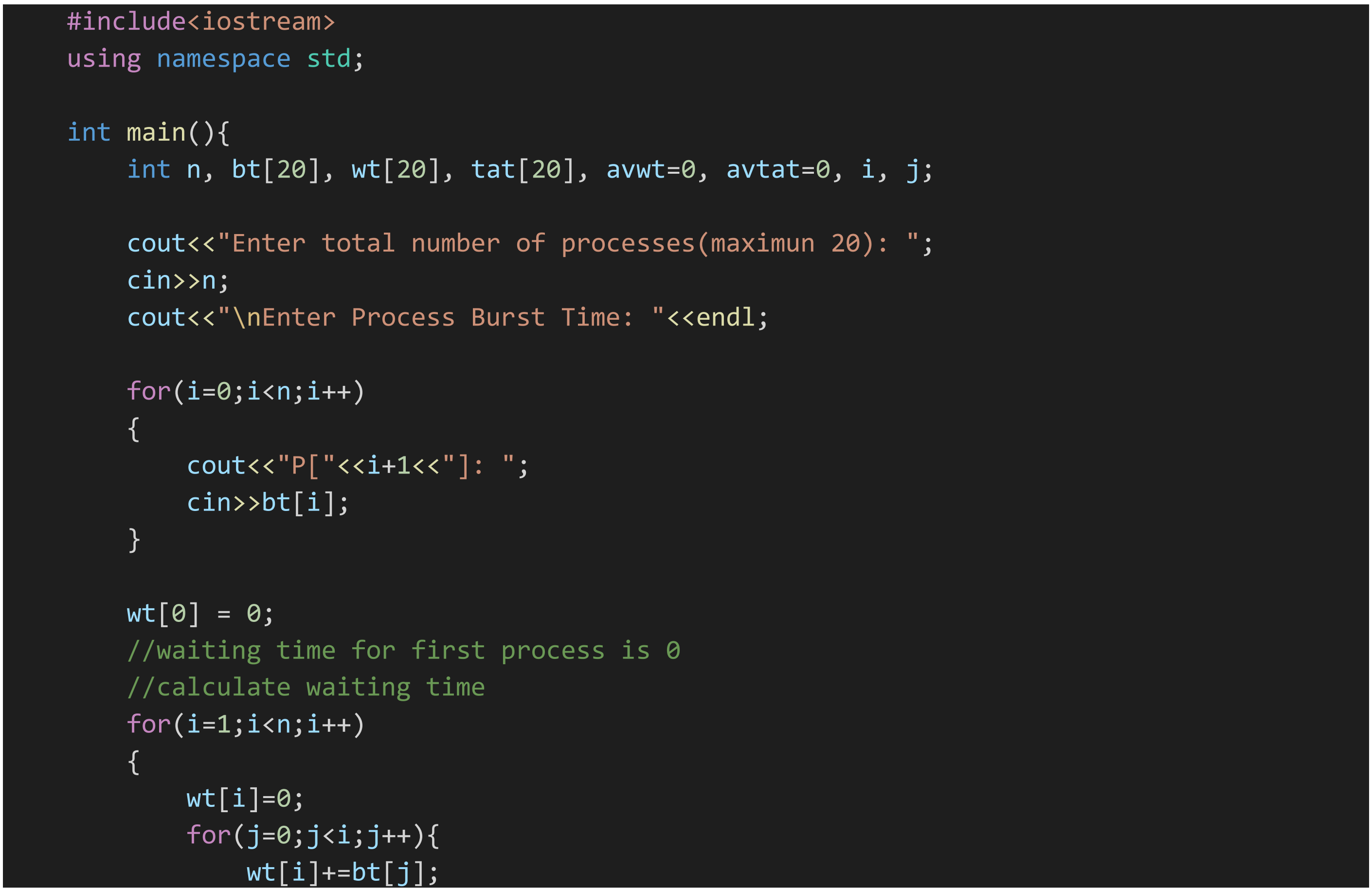
```
#include<stdio.h>
#include<iostream>
#include<fstream>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>

using namespace std;

void copy(int old,int new1);

int main(int argc,char* (argv[])){
    int fdold,fdnew;

    if(argc != 3){
        printf("Needs two arguments\n");
        exit(0);
    }

    fdold = open(argv[1],0);

    if(fdold == -1){
        printf("Unable to open file\n");
        exit(0);
    }

    fdnew = creat(argv[2],6666);

    if(fdnew == -1){
        printf("Unable to create file\n");
        exit(0);
    }

    copy(fdold,fdnew);
    return 0;
}

void copy(int old,int new1){
    int count;
    char buffer[512];
    while((count=read(old,buffer,sizeof(buffer)))>0){
        write(new1,buffer,count);
    }
}
```

OUTPUT:



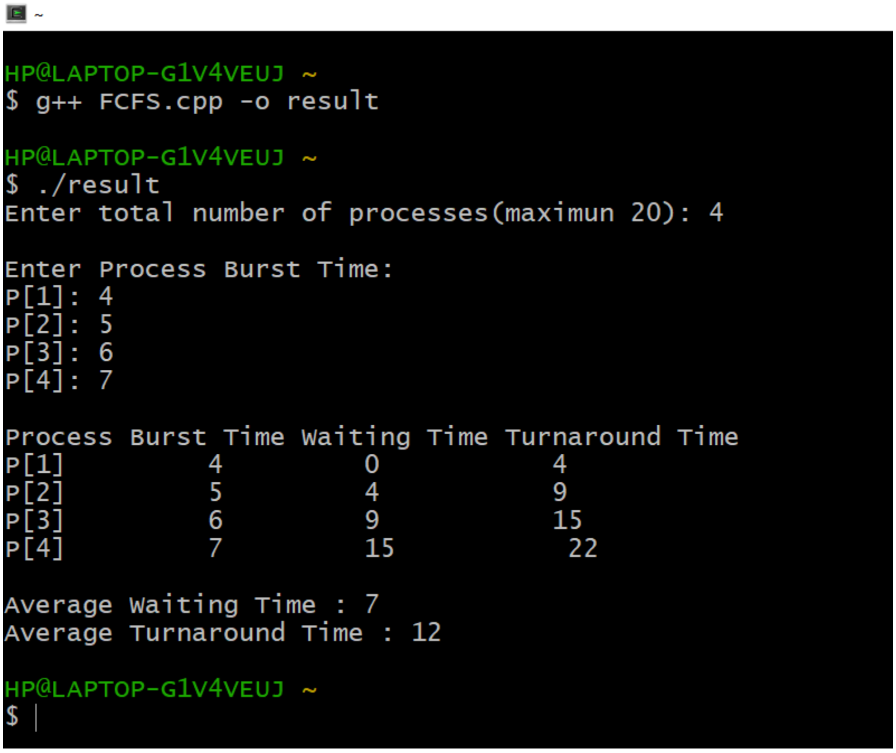
```
~
HP@LAPTOP-G1V4VEUJ ~
$ touch f1.txt f2.txt

HP@LAPTOP-G1V4VEUJ ~
$ cat>f1.txt
Hello World!

HP@LAPTOP-G1V4VEUJ ~
$ g++ Q5.cpp -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res
Needs two arguments

HP@LAPTOP-G1V4VEUJ ~
$ ./res f1.txt
Needs two arguments

HP@LAPTOP-G1V4VEUJ ~
$ ./res f1.txt f2.txt

HP@LAPTOP-G1V4VEUJ ~
$ cat f2.txt
Hello World!

HP@LAPTOP-G1V4VEUJ ~
$ |
```

Q6. Write a program to implement FCFS scheduling algorithm.



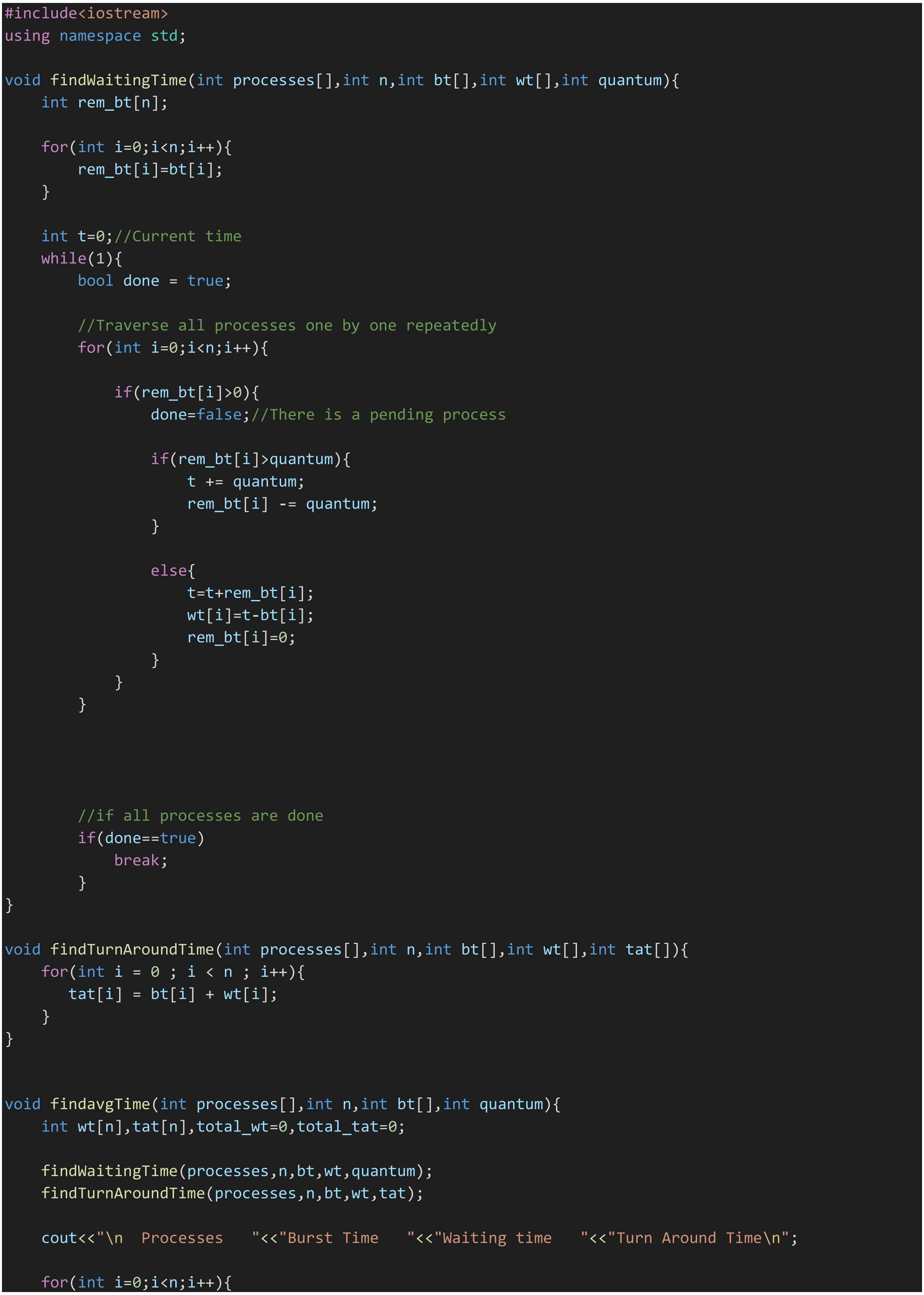
```
#include<iostream>
using namespace std;

int main(){
    int n, bt[20], wt[20], tat[20], avwt=0, avtat=0, i, j;

    cout<<"Enter total number of processes(maximun 20): ";
    cin>>n;
    cout<<"\nEnter Process Burst Time: "<<endl;

    for(i=0;i<n;i++)
    {
        cout<<"P["<<i+1<<"]: ";
        cin>>bt[i];
    }

    wt[0] = 0;
    //waiting time for first process is 0
    //calculate waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++){
            wt[i]+=bt[j];
        }
    }

    cout<<"\nProcess Burst Time Waiting Time Turnaround Time";
    //calculating turnaround time
    for(i=0;i<n;i++){
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
        cout<<endl<<"P["<<i+1<<"]"<<"    "<<bt[i]<<"    "<<wt[i]<<"   "<<tat[i];
    }

    avwt/=i;
    avtat/=i;
    cout<<endl;
    cout<<"\nAverage Waiting Time : "<<avwt;
    cout<<endl;
    cout<<"Average Turnaround Time : "<<avtat;
    cout<<endl;
    return 0;
}
```

OUTPUT:

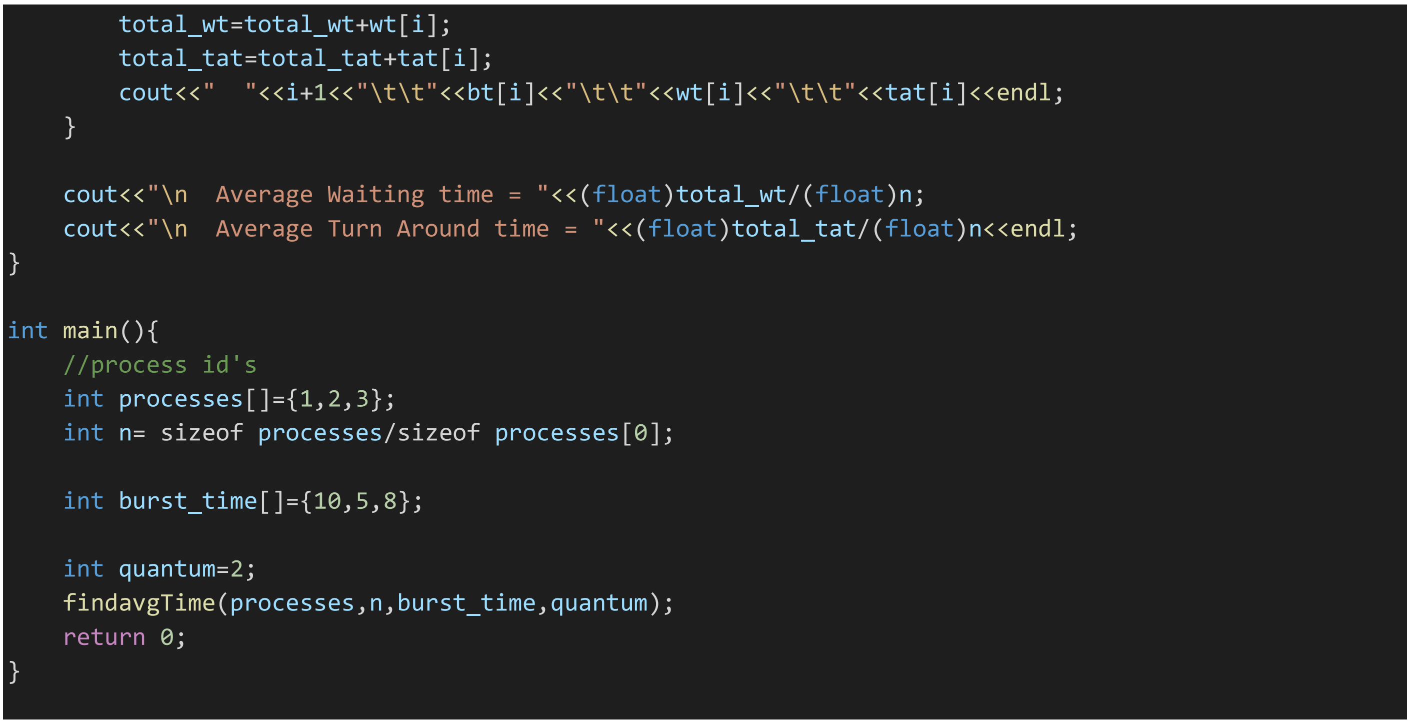


```
~
HP@LAPTOP-G1V4VEUJ ~
$ g++ FCFS.cpp -o result

HP@LAPTOP-G1V4VEUJ ~
$ ./result
Enter total number of processes(maximun 20): 4

Enter Process Burst Time:
P[1]: 4
P[2]: 5
P[3]: 6
P[4]: 7

Process Burst Time Waiting Time Turnaround Time
P[1]         4         0           4
P[2]         5         4           9
P[3]         6         9           15
P[4]         7         15           22

Average Waiting Time : 7
Average Turnaround Time : 12

HP@LAPTOP-G1V4VEUJ ~
$
```

Q7. Write a program to implement Round Robin scheduling algorithm.

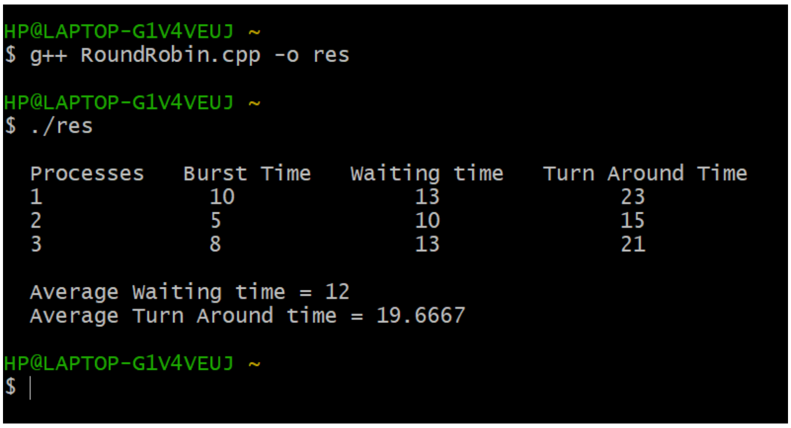


```
#include<iostream>
using namespace std;

void findWaitingTime(int processes[],int n,int bt[],int wt[],int quantum){
    int rem_bt[n];

    for(int i=0;i<n;i++){
        rem_bt[i]=bt[i];
    }

    int t=0;//Current time
    while(1){
        bool done = true;

        //Traverse all processes one by one repeatedly
        for(int i=0;i<n;i++){

            if(rem_bt[i]>0){
                done=false;//There is a pending process

                if(rem_bt[i]>quantum){
                    t += quantum;
                    rem_bt[i] -= quantum;
                }

                else{
                    t=t+rem_bt[i];
                    wt[i]=t-bt[i];
                    rem_bt[i]=0;
                }
            }
        }




        //if all processes are done
        if(done==true)
            break;
        }
}

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 quantum){
    int wt[n],tat[n],total_wt=0,total_tat=0;

    findWaitingTime(processes,n,bt,wt,quantum);
    findTurnAroundTime(processes,n,bt,wt,tat);

    cout<<"\n  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];
        cout<<"  "<<i+1<<"\t\t"<<bt[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i]<<endl;
    }

    cout<<"\n  Average Waiting time = "<<(float)total_wt/(float)n;
    cout<<"\n  Average Turn Around time = "<<(float)total_tat/(float)n<<endl;
}

int main(){
    //process id's
    int processes[]={1,2,3};
    int n= sizeof processes/sizeof processes[0];

    int burst_time[]={10,5,8};

    int quantum=2;
    findavgTime(processes,n,burst_time,quantum);
    return 0;
}
```

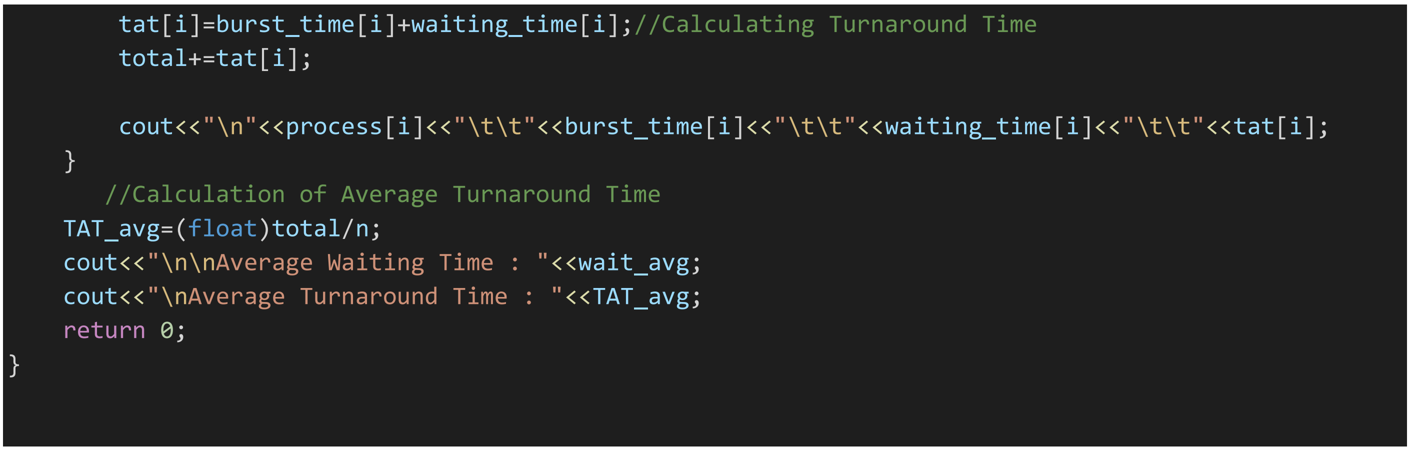
OUTPUT:



```
HP@LAPTOP-G1V4VEUJ ~
$ g++ RoundRobin.cpp -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

  Processes   Burst Time   Waiting time   Turn Around Time
  1             10              13              23
  2             5               10              15
  3             8               13              21

  Average Waiting time = 12
  Average Turn Around time = 19.6667

HP@LAPTOP-G1V4VEUJ ~
$ |
```

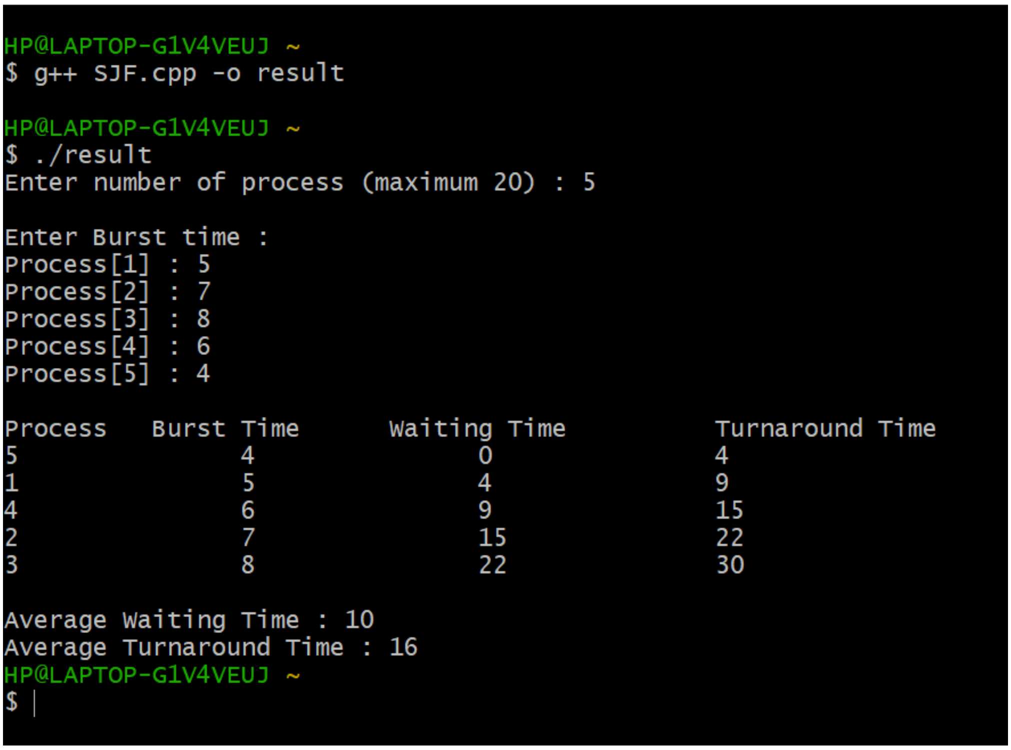
Q8. Write a program to implement SJF scheduling algorithm.

```
#include<iostream>
using namespace std;

int main(){

    int burst_time[20],process[20],waiting_time[20],tat[2],i,j,n,total=0,pos,temp;
    float wait_avg,TAT_avg;

    cout<<"Enter number of process (maximum 20) : ";
    cin>>n;
    cout<<"\nEnter Burst time : \n";

    for(i=0;i<n;i++){
        cout<<"Process["<<i+1<<"] : ";
        cin>>burst_time[i];
        process[i] = i+1;    //Process Number
    }

    //Sorting
    for(i=0;i<n;i++){
        pos=i;

        for(j=i+1;j<n;j++){

            if(burst_time[j]<burst_time[pos]){
                pos=j;
            }
        }

        temp=burst_time[i];
        burst_time[i]=burst_time[pos];
        burst_time[pos]=temp;

        temp=process[i];
        process[i]=process[pos];
        process[pos]=temp;
    }

    //First process has 0 waiting time
    waiting_time[0]=0;

    //Calculating waiting time
    for(i=1;i<n;i++){
        waiting_time[i]=0;

        for(j=0;j<i;j++){
            waiting_time[i]+=burst_time[j];
        }

        total+=waiting_time[i];
    }

    //Calculating Average waiting time
    wait_avg=(float)total/n;
    total=0;

    cout<<"\nProcess\t  Burst Time  \t  Waiting Time  \tTurnaround Time  ";
    for(i=0;i<n;i++){
```



```
        tat[i]=burst_time[i]+waiting_time[i];//Calculating Turnaround Time
        total+=tat[i];

        cout<<"\n"<<process[i]<<"\t\t"<<burst_time[i]<<"\t\t"<<waiting_time[i]<<"\t\t"<<tat[i];
    }
       //Calculation of Average Turnaround Time
    TAT_avg=(float)total/n;
    cout<<"\n\nAverage Waiting Time : "<<wait_avg;
    cout<<"\nAverage Turnaround Time : "<<TAT_avg;
    return 0;
}
```

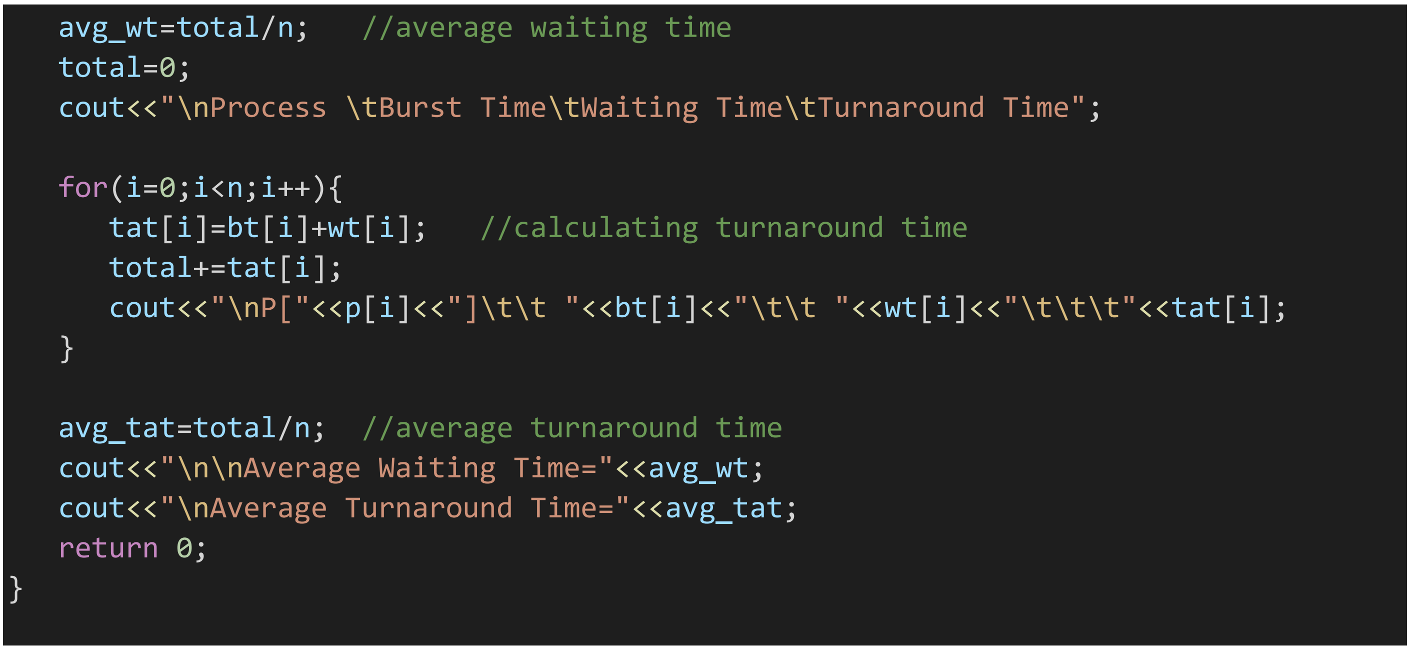
OUTPUT:



```
HP@LAPTOP-G1V4VEUJ ~
$ g++ SJF.cpp -o result

HP@LAPTOP-G1V4VEUJ ~
$ ./result
Enter number of process (maximum 20) : 5

Enter Burst time :
Process[1] : 5
Process[2] : 7
Process[3] : 8
Process[4] : 6
Process[5] : 4

Process   Burst Time      Waiting Time          Turnaround Time
5               4               0               4
1               5               4               9
4               6               9               15
2               7               15              22
3               8               22              30

Average Waiting Time : 10
Average Turnaround Time : 16
HP@LAPTOP-G1V4VEUJ ~
$ |
```

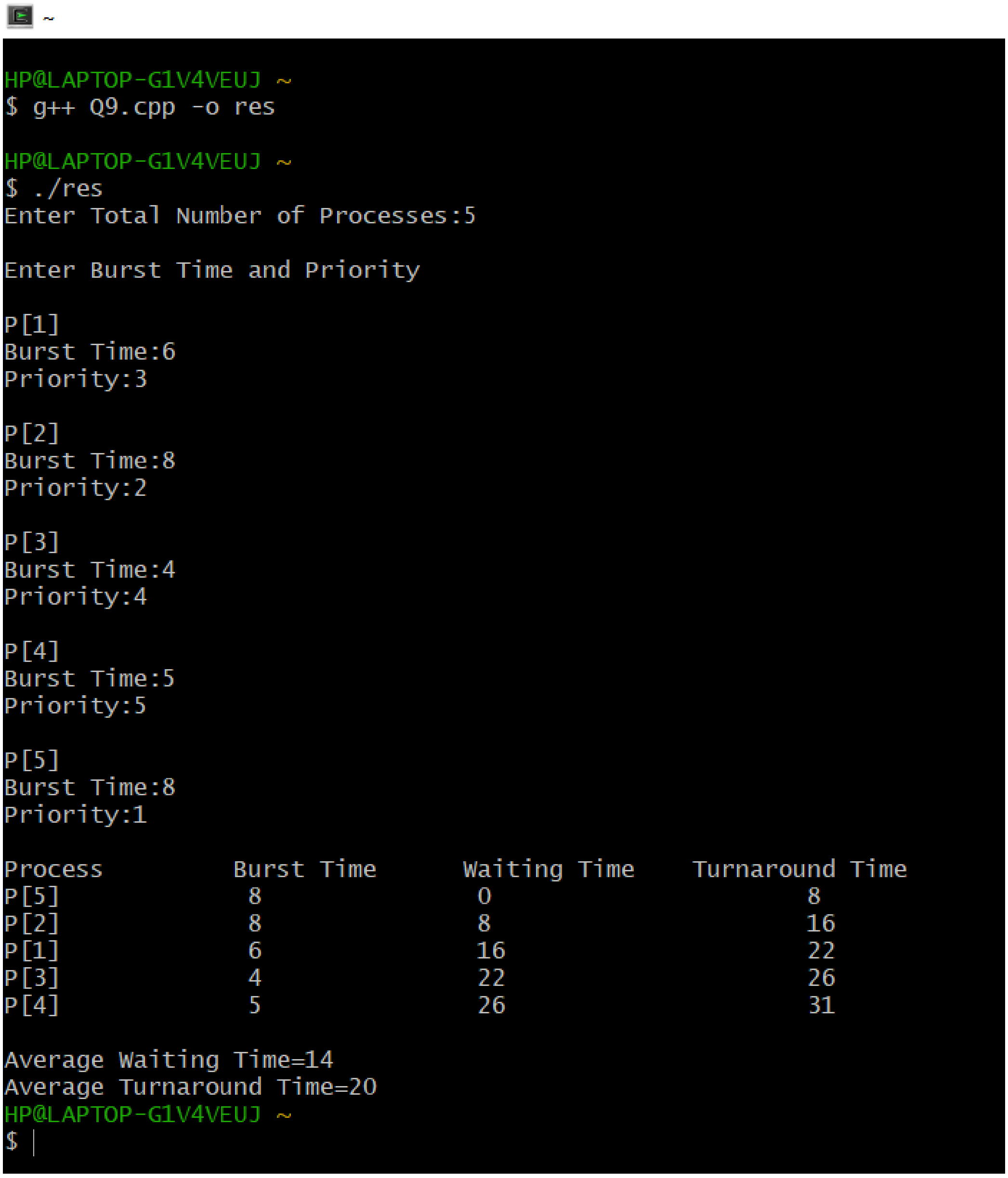
Q9. Write a program to implement non pre-emptive priority-based scheduling algorithm.

```cpp
#include<iostream>
using namespace std;

int main(){
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   cout<<"Enter Total Number of Processes:";
   cin>>n;
   cout<<"\nEnter Burst Time and Priority\n";

   for(i=0;i<n;i++){
      cout<<"\nP["<<i+1<<"]\n";
      cout<<"Burst Time:";
      cin>>bt[i];

      cout<<"Priority:";
      cin>>pr[i];
      p[i]=i+1;   //contains process number
   }

   //sorting burst time,priority and process number in ascending order using selection sort
   for(i=0;i<n;i++){
      pos=i;

      for(j=i+1;j<n;j++){
         if(pr[j]<pr[pos])
            pos=j;
      }

      temp=pr[i];
      pr[i]=bt[pos];
      pr[pos]=temp;
      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;
      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }

   wt[0]=0;
   //calculate waiting time
   for(i=1;i<n;i++){
      wt[i]=0;
      for(j=0;j<i;j++)
         wt[i]+=bt[j];
      total+=wt[i];
   }
```



```
    avg_wt=total/n;    //average waiting time
    total=0;
    cout<<"\nProcess \tBurst Time\tWaiting Time\tTurnaround Time";

    for(i=0;i<n;i++){
        tat[i]=bt[i]+wt[i];    //calculating turnaround time
        total+=tat[i];
        cout<<"\nP["<<p[i]<<"]\t\t "<<bt[i]<<"\t\t "<<wt[i]<<"\t\t\t"<<tat[i];
    }

    avg_tat=total/n;  //average turnaround time
    cout<<"\n\nAverage Waiting Time="<<avg_wt;
    cout<<"\nAverage Turnaround Time="<<avg_tat;
    return 0;
}
```

OUTPUT:



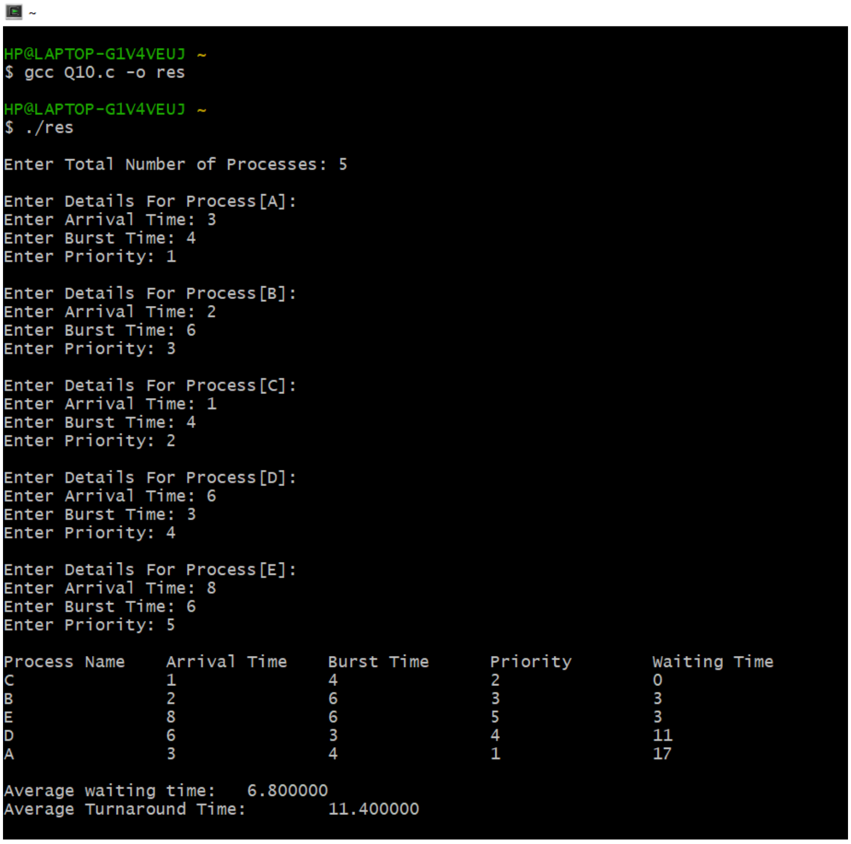
```
~

HP@LAPTOP-G1V4VEUJ ~
$ g++ Q9.cpp -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res
Enter Total Number of Processes:5

Enter Burst Time and Priority

P[1]
Burst Time:6
Priority:3

P[2]
Burst Time:8
Priority:2

P[3]
Burst Time:4
Priority:4

P[4]
Burst Time:5
Priority:5

P[5]
Burst Time:8
Priority:1

Process         Burst Time      Waiting Time    Turnaround Time
P[5]             8               0                       8
P[2]             8               8                       16
P[1]             6               16                      22
P[3]             4               22                      26
P[4]             5               26                      31

Average Waiting Time=14
Average Turnaround Time=20
HP@LAPTOP-G1V4VEUJ ~
$ 
```

Q10. Write a program to implement pre-emptive priority-based scheduling algorithm.

```
#include<stdio.h>

struct process{
  char process_name;
  int arrival_time,burst_time,ct,waiting_time,turnaround_time,priority;
  int status;
}

process_queue[10];
int limit;

void Arrival_Time_Sorting(){
  struct process temp;
  int i,j;

  for(i=0;i<limit-1;i++){
    for(j=i+1;j<limit;j++){
      if(process_queue[i].arrival_time>process_queue[j].arrival_time){
        temp=process_queue[i];
        process_queue[i]=process_queue[j];
        process_queue[j]=temp;
      }
    }
  }
}

void main(){
  int i,time=0,burst_time=0,largest;
  char c;
  float wait_time=0,turnaround_time=0,average_waiting_time,average_turnaround_time;

  printf("\nEnter Total Number of Processes: ");
  scanf("%d",&limit);

  for(i=0,c='A';i<limit;i++,c++){
    process_queue[i].process_name=c;
    printf("\nEnter Details For Process[%C]:\n",process_queue[i].process_name);
    printf("Enter Arrival Time: ");
    scanf("%d",&process_queue[i].arrival_time);
    printf("Enter Burst Time: ");
    scanf("%d",&process_queue[i].burst_time);
    printf("Enter Priority: ");
    scanf("%d",&process_queue[i].priority);
    process_queue[i].status=0;
    burst_time=burst_time+process_queue[i].burst_time;
  }

  Arrival_Time_Sorting();
  process_queue[9].priority=-9999;
```

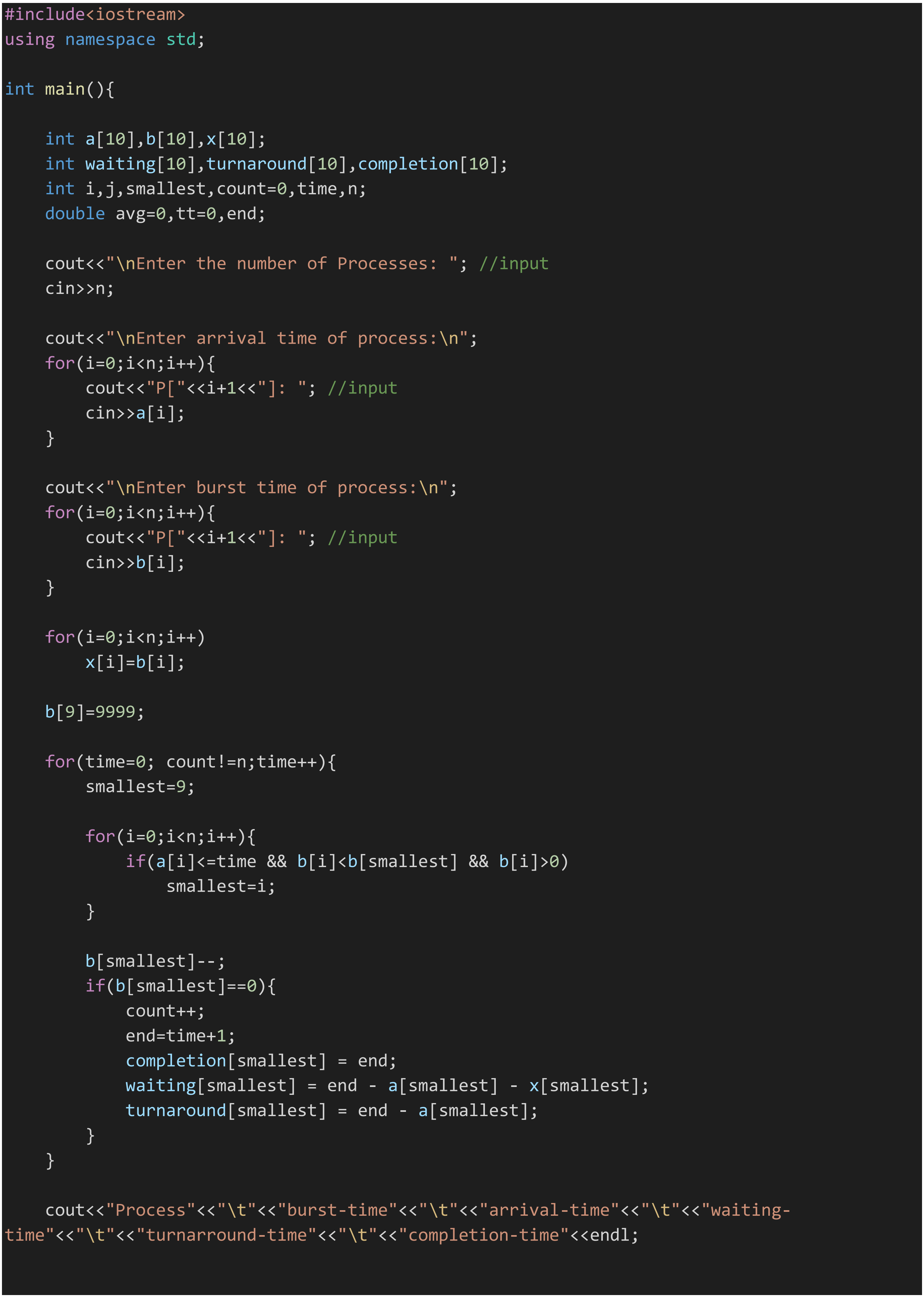
```c
  printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting Time");
  for(time=process_queue[0].arrival_time;time<burst_time;){
    largest=9;
    for(i=0;i<limit;i++){
      if(process_queue[i].arrival_time<=time&&process_queue[i].status!=1&&process_queue[i].priority>process_queue[largest].priority){
        largest=i;
      }
    }

    time=time+process_queue[largest].burst_time;
    process_queue[largest].ct=time;
    process_queue[largest].waiting_time=process_queue[largest].ct-process_queue[largest].arrival_time-process_queue[largest].burst_time;
    process_queue[largest].turnaround_time=process_queue[largest].ct-process_queue[largest].arrival_time;
    process_queue[largest].status=1;
    wait_time=wait_time+process_queue[largest].waiting_time;
    turnaround_time=turnaround_time+process_queue[largest].turnaround_time;
    printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d",process_queue[largest].process_name,process_queue[largest].arrival_time,process_queue[largest].burst_time,process_queue[largest].priority,process_queue[largest].waiting_time);
  }

  average_waiting_time=wait_time/limit;
  average_turnaround_time=turnaround_time/limit;
  printf("\n\nAverage waiting time:\t%f\n",average_waiting_time);
  printf("Average Turnaround Time:\t%f\n",average_turnaround_time);
}
```

OUTPUT:



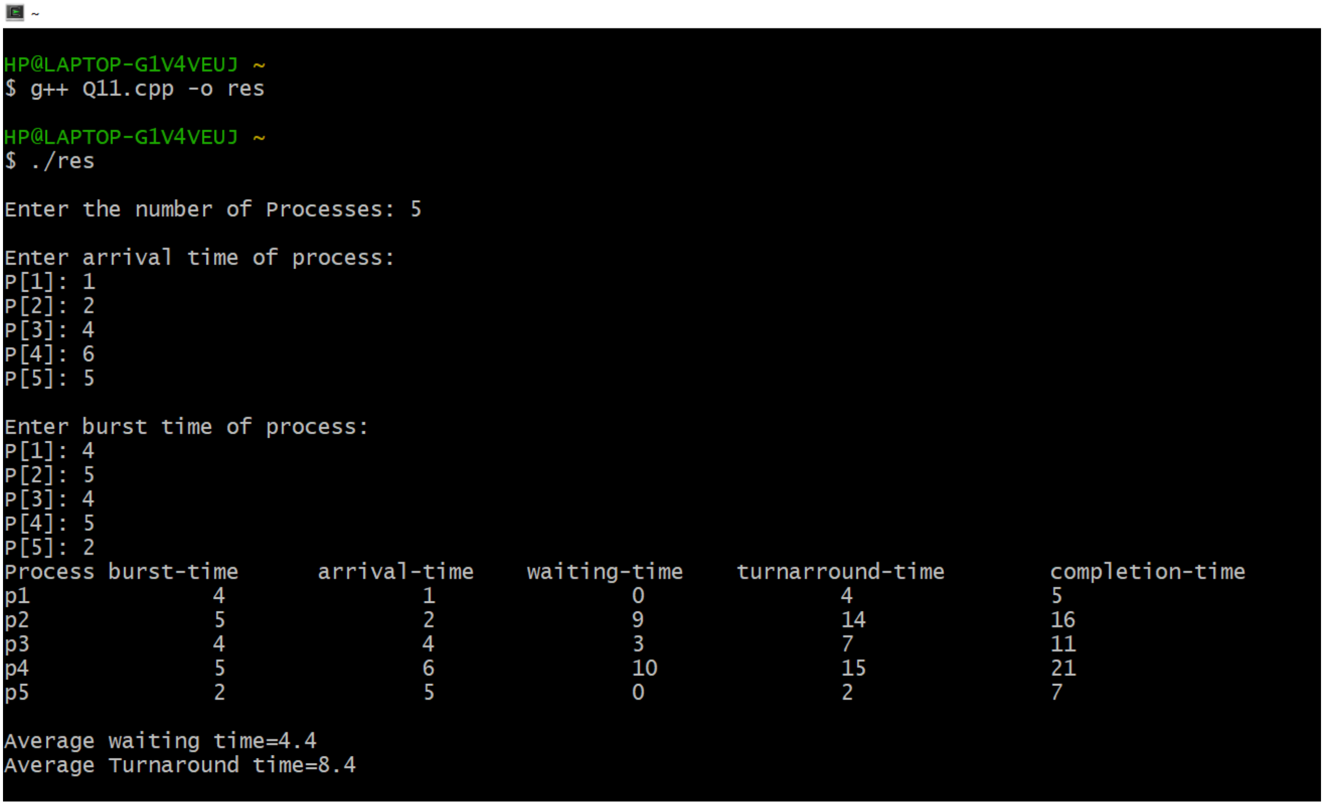
```
HP@LAPTOP-G1V4VEUJ ~
$ gcc Q10.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

Enter Total Number of Processes: 5

Enter Details For Process[A]:
Enter Arrival Time: 3
Enter Burst Time: 4
Enter Priority: 1

Enter Details For Process[B]:
Enter Arrival Time: 2
Enter Burst Time: 6
Enter Priority: 3

Enter Details For Process[C]:
Enter Arrival Time: 1
Enter Burst Time: 4
Enter Priority: 2

Enter Details For Process[D]:
Enter Arrival Time: 6
Enter Burst Time: 3
Enter Priority: 4

Enter Details For Process[E]:
Enter Arrival Time: 8
Enter Burst Time: 6
Enter Priority: 5

Process Name    Arrival Time    Burst Time      Priority        Waiting Time
C               1               4               2               0
B               2               6               3               3
E               8               6               5               3
D               6               3               4               11
A               3               4               1               17

Average waiting time:   6.800000
Average Turnaround Time:        11.400000
```

Q11. Write a program to implement SRJF scheduling algorithm.



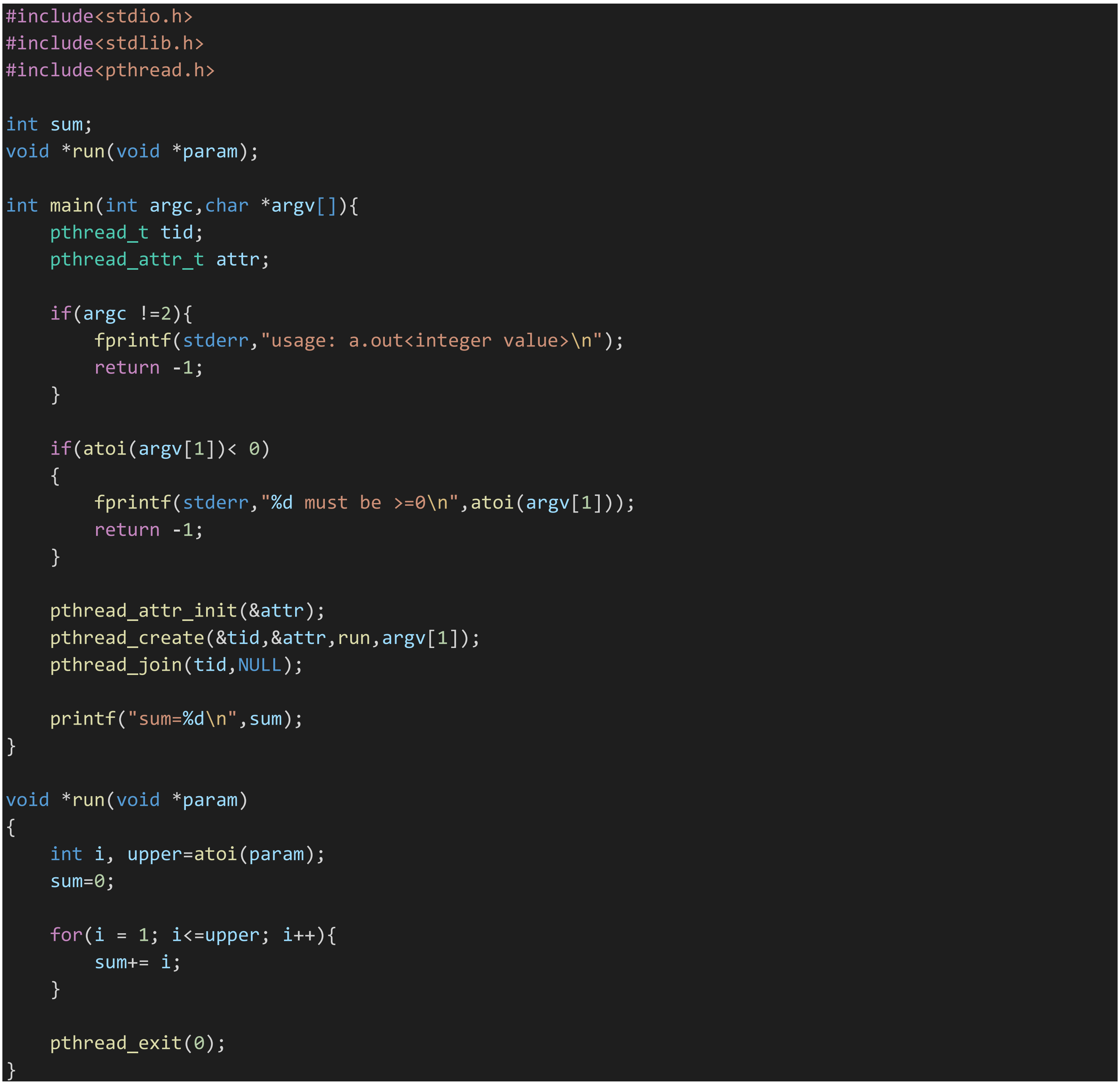
```
#include<iostream>
using namespace std;

int main(){

    int a[10],b[10],x[10];
    int waiting[10],turnaround[10],completion[10];
    int i,j,smallest,count=0,time,n;
    double avg=0,tt=0,end;

    cout<<"\nEnter the number of Processes: "; //input
    cin>>n;

    cout<<"\nEnter arrival time of process:\n";
    for(i=0;i<n;i++){
        cout<<"P["<<i+1<<"]: "; //input
        cin>>a[i];
    }

    cout<<"\nEnter burst time of process:\n";
    for(i=0;i<n;i++){
        cout<<"P["<<i+1<<"]: "; //input
        cin>>b[i];
    }

    for(i=0;i<n;i++)
        x[i]=b[i];

    b[9]=9999;

    for(time=0; count!=n;time++){
        smallest=9;

        for(i=0;i<n;i++){
            if(a[i]<=time && b[i]<b[smallest] && b[i]>0)
                smallest=i;
        }

        b[smallest]--;
        if(b[smallest]==0){
            count++;
            end=time+1;
            completion[smallest] = end;
            waiting[smallest] = end - a[smallest] - x[smallest];
            turnaround[smallest] = end - a[smallest];
        }
    }

    cout<<"Process"<<"\t"<<"burst-time"<<"\t"<<"arrival-time"<<"\t"<<"waiting-time"<<"\t"<<"turnarround-time"<<"\t"<<"completion-time"<<endl;
    for(i=0;i<n;i++){
        cout<<"p"<<i+1<<"\t\t"<<x[i]<<"\t\t"<<a[i]<<"\t\t"<<waiting[i]<<"\t\t"<<turnaround
[i]<<"\t\t"<<completion[i]<<endl;
        avg=avg+waiting[i];
        tt=tt+turnaround[i];
    }

    cout<<"\nAverage waiting time="<<avg/n;
    cout<<"\nAverage Turnaround time="<<tt/n<<endl;
    return 0;
}
```

OUTPUT:



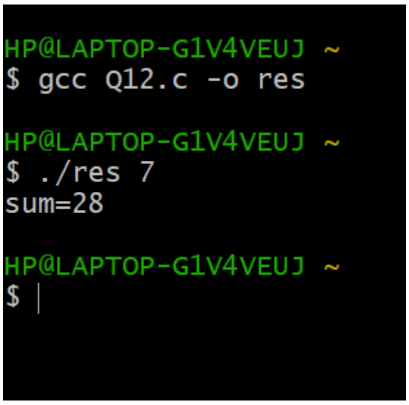
```
~
HP@LAPTOP-G1V4VEUJ ~
$ g++ Q11.cpp -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

Enter the number of Processes: 5

Enter arrival time of process:
P[1]: 1
P[2]: 2
P[3]: 4
P[4]: 6
P[5]: 5

Enter burst time of process:
P[1]: 4
P[2]: 5
P[3]: 4
P[4]: 5
P[5]: 2
Process burst-time      arrival-time    waiting-time    turnarround-time        completion-time
p1              4               1               0               4               5
p2              5               2               9               14              16
p3              4               4               3               7               11
p4              5               6               10              15              21
p5              2               5               0               2               7

Average waiting time=4.4
Average Turnaround time=8.4
```

Q12. Write a program to calculate sum of n numbers using thread library.



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

int sum;
void *run(void *param);

int main(int argc,char *argv[]){
    pthread_t tid;
    pthread_attr_t attr;

    if(argc !=2){
        fprintf(stderr,"usage: a.out<integer value>\n");
        return -1;
    }

    if(atoi(argv[1])< 0)
    {
        fprintf(stderr,"%d must be >=0\n",atoi(argv[1]));
        return -1;
    }

    pthread_attr_init(&attr);
    pthread_create(&tid,&attr,run,argv[1]);
    pthread_join(tid,NULL);

    printf("sum=%d\n",sum);
}

void *run(void *param)
{
    int i, upper=atoi(param);
    sum=0;

    for(i = 1; i<=upper; i++){
        sum+= i;
    }

    pthread_exit(0);
}
```

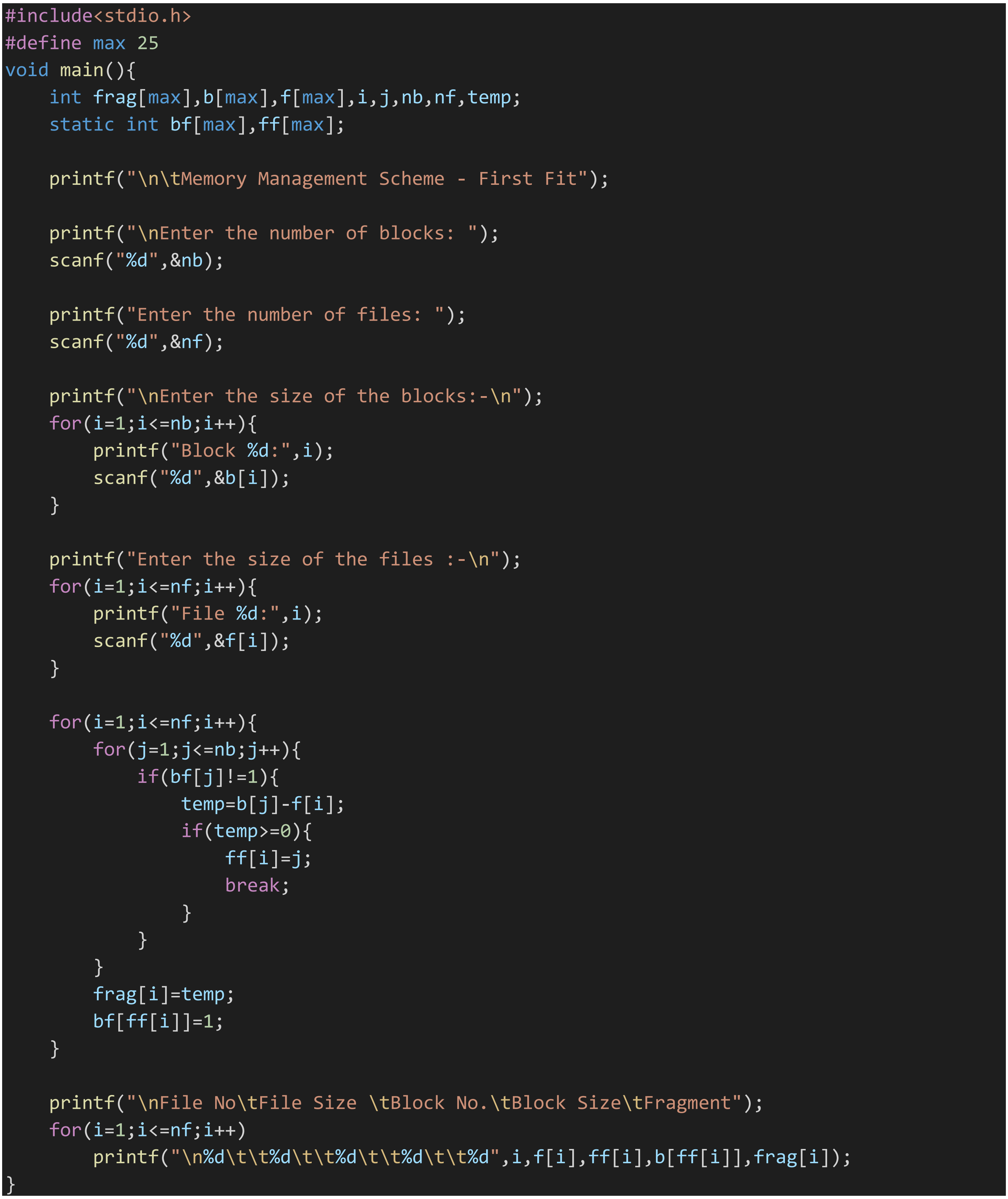
OUTPUT:



```
HP@LAPTOP-G1V4VEUJ ~
$ gcc Q12.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res 7
sum=28

HP@LAPTOP-G1V4VEUJ ~
$ 
```

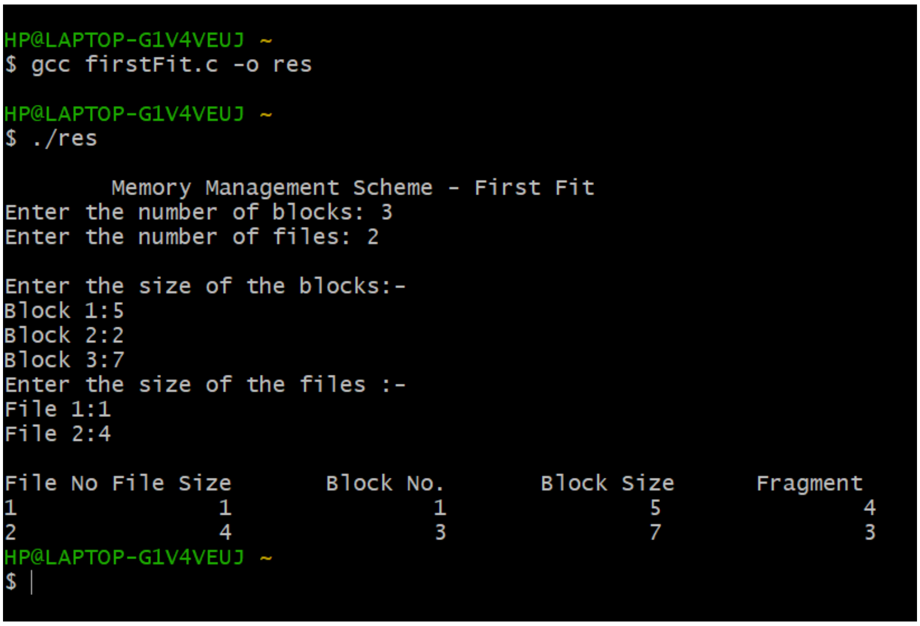
Q13. Write a program to implement first-fit, best-fit, and worst-fit allocation strategies.

FIRST-FIT



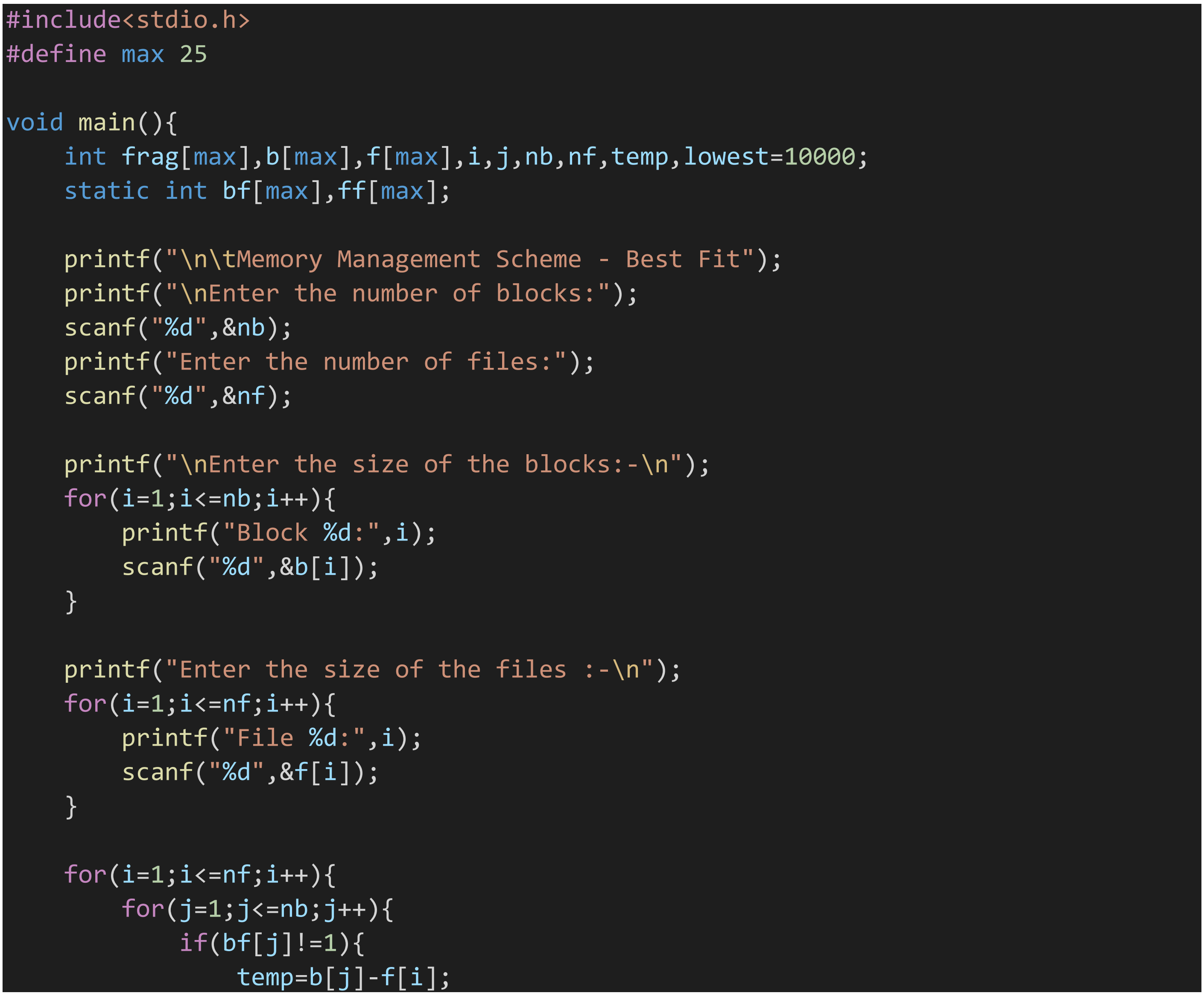
```
#include<stdio.h>
#define max 25
void main(){
    int frag[max],b[max],f[max],i,j,nb,nf,temp;
    static int bf[max],ff[max];

    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.\tBlock Size\tFragment");
    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]);
}
```

OUTPUT:

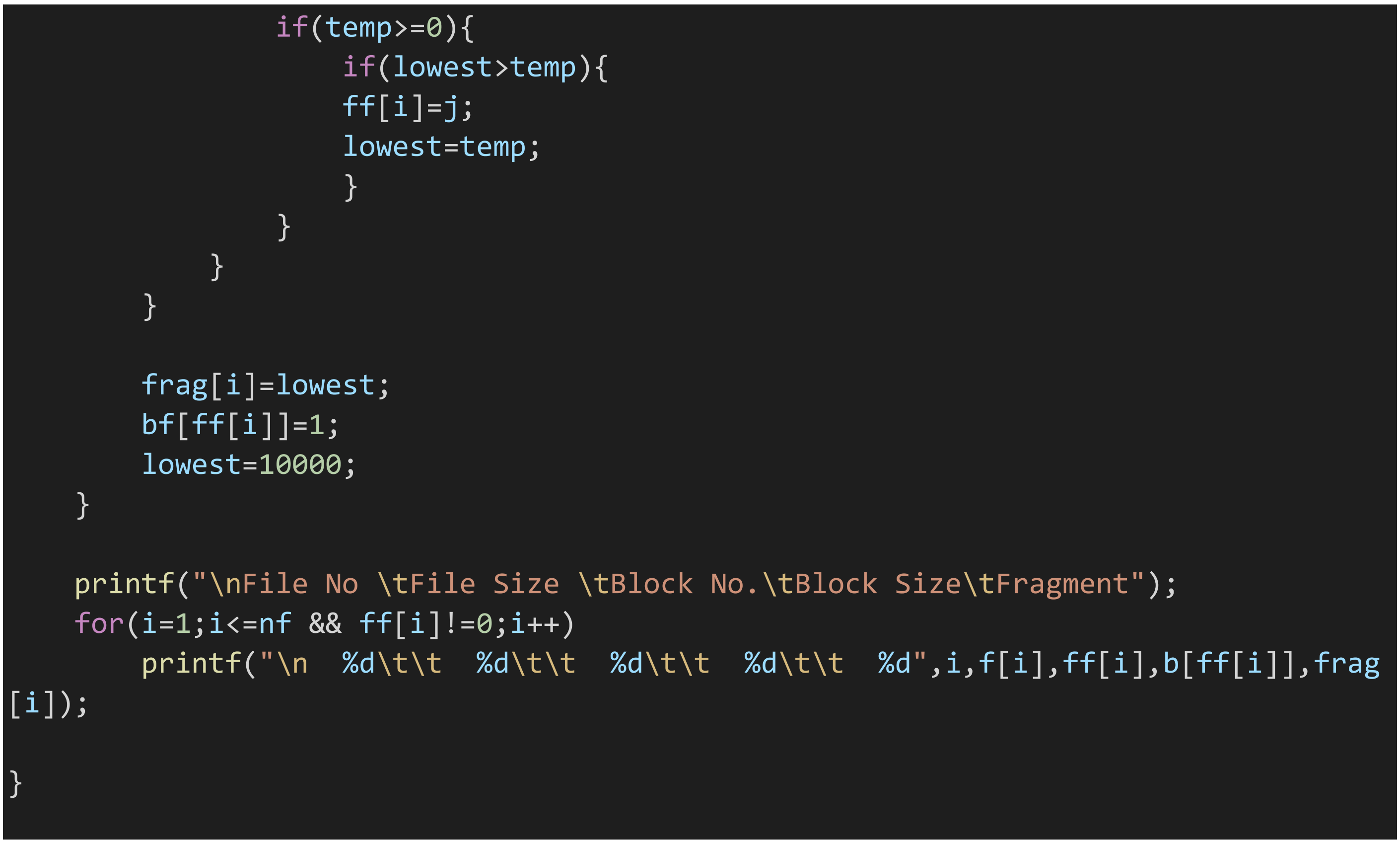


```
HP@LAPTOP-G1V4VEUJ ~
$ gcc firstFit.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

        Memory Management Scheme - First Fit
Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File No File Size       Block No.       Block Size      Fragment
1               1               1               5               4
2               4               3               7               3
HP@LAPTOP-G1V4VEUJ ~
$ 
```

BEST-FIT

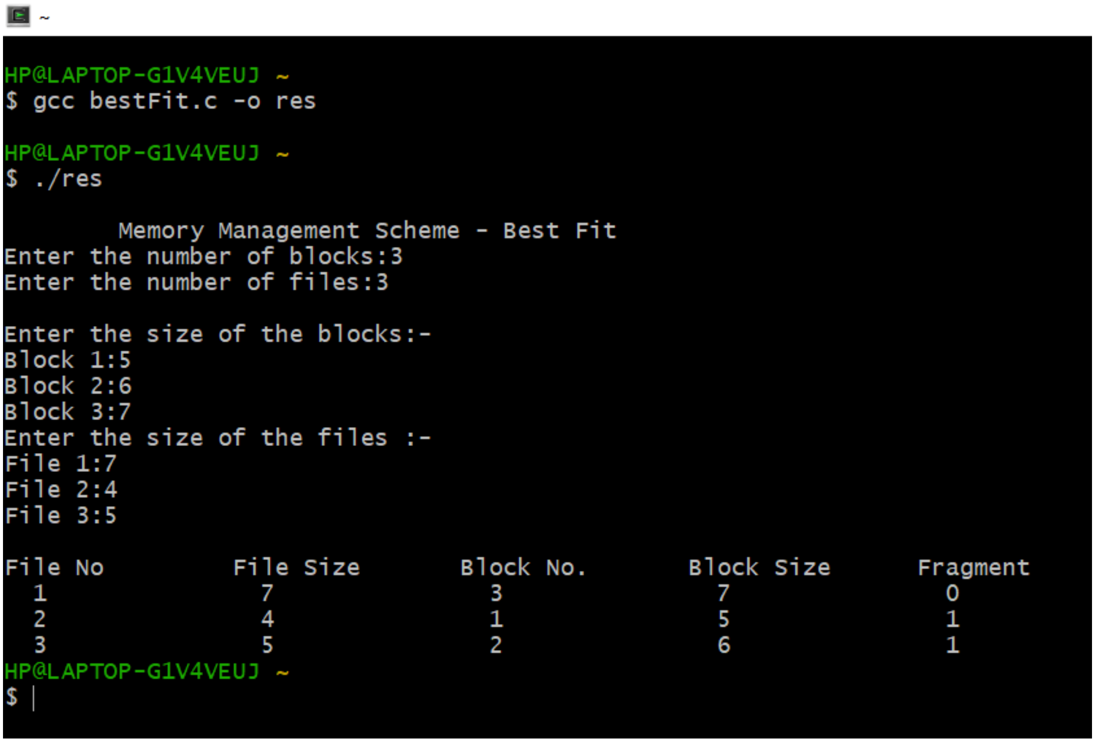


```c
#include<stdio.h>
#define max 25

void main(){
    int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
    static int bf[max],ff[max];

    printf("\n\tMemory Management Scheme - Best 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];
```



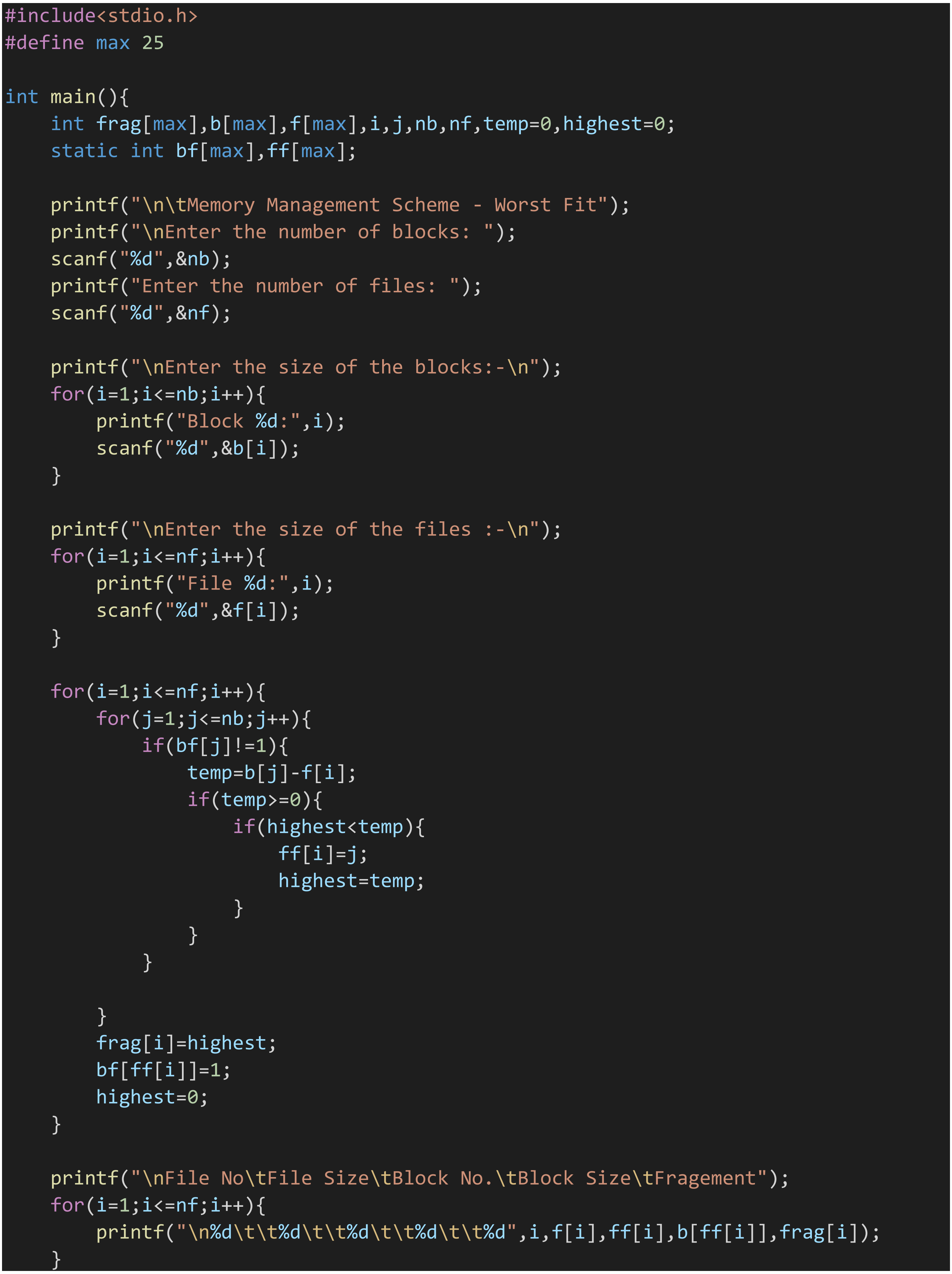
```
        if(temp>=0){
          if(lowest>temp){
          ff[i]=j;
          lowest=temp;
          }
        }
      }
    }

    frag[i]=lowest;
    bf[ff[i]]=1;
    lowest=10000;
  }

  printf("\nFile No \tFile Size \tBlock No.\tBlock Size\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]);

}
```

OUTPUT:



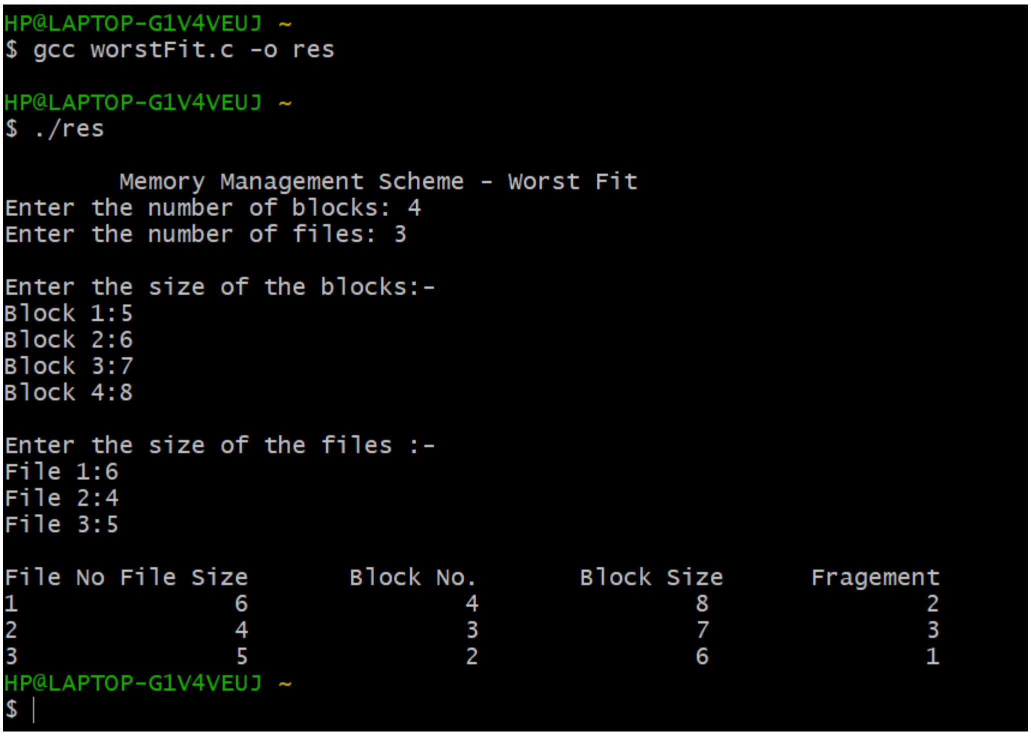
```
~
HP@LAPTOP-G1V4VEUJ ~
$ gcc bestFit.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

        Memory Management Scheme - Best Fit
Enter the number of blocks:3
Enter the number of files:3

Enter the size of the blocks:-
Block 1:5
Block 2:6
Block 3:7
Enter the size of the files :-
File 1:7
File 2:4
File 3:5

File No         File Size       Block No.       Block Size      Fragment
  1               7               3               7               0
  2               4               1               5               1
  3               5               2               6               1
HP@LAPTOP-G1V4VEUJ ~
$ |
```

WORST-FIT



```
#include<stdio.h>
#define max 25

int main(){
    int frag[max],b[max],f[max],i,j,nb,nf,temp=0,highest=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("\nEnter 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){
                        ff[i]=j;
                        highest=temp;
                    }
                }
            }

        }
        frag[i]=highest;
        bf[ff[i]]=1;
        highest=0;
    }

    printf("\nFile No\tFile Size\tBlock No.\tBlock 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]);
    }
    return 0;

}
```

OUTPUT:



```
HP@LAPTOP-G1V4VEUJ ~
$ gcc worstFit.c -o res

HP@LAPTOP-G1V4VEUJ ~
$ ./res

        Memory Management Scheme - Worst Fit
Enter the number of blocks: 4
Enter the number of files: 3

Enter the size of the blocks:-
Block 1:5
Block 2:6
Block 3:7
Block 4:8

Enter the size of the files :-
File 1:6
File 2:4
File 3:5

File No File Size       Block No.       Block Size      Fragement
1               6               4               8               2
2               4               3               7               3
3               5               2               6               1
HP@LAPTOP-G1V4VEUJ ~
$ 
```