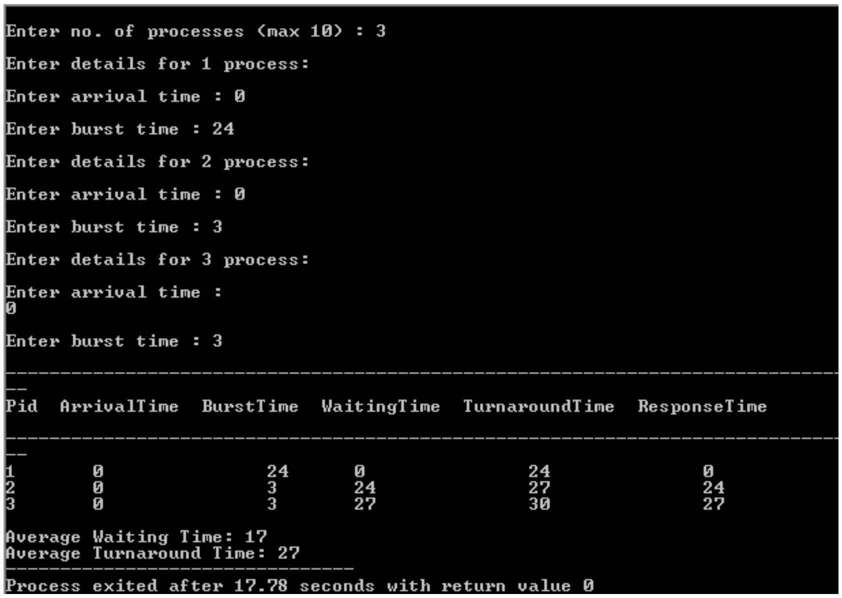
6. Write a program to implement FCFS scheduling algorithm.

A6.

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

class Process
{
      float at;
      float bt;
      float wt;
      float tar;
      float res;
      float avgwt;
      float avgtt;

      public:
            void entry()
            {
                  cout<<"\nEnter arrival time : ";
                  cin>>at;

                  cout<<"\nEnter burst time : ";
                  cin>>bt;
            }

            void sorter(Process p[], int size)
            {
                  for(int i=1; i<=size; ++i)
                  {
                        for(int j=1; j<=size-1; ++j)
                        {
                              if(p[j].at>p[j+1].at)
                              {
                                    Process temp=p[j];
                                    p[j]=p[j+1];
                                    p[j+1]=temp;
                              }
                        }
                  }
            }

            void cal_wait(Process p[], int sz)
            {
                  avgwt=0;
                  p[1].wt=0;
                  for(int a=2; a<=sz; ++a)
                  {
                        p[a].wt=0;
                        for(int b=1; b<a; ++b)
                              p[a].wt+=p[b].bt;
                        p[a].wt-=p[a].at;
                  }
                  for(int i=1; i<=sz; i++)
                        avgwt+=p[i].wt;
                  avgwt=avgwt/sz;
            }

            void cal_trn(Process p[], int s)
            {
                  avgtt=0;
                  for(int i=1; i<=s; ++i)
                        p[i].tar=p[i].wt+p[i].bt;
                  for(int i=1; i<=s; i++)
                        avgtt+=p[i].tar;
                  avgtt=avgtt/s;
            }

            void cal_res(Process p[], int y)
            {
                  p[1].res=0;
                  for(int i=2; i<=y; ++i)
                  {
                        p[i].res=0;
                        for(int j=2; j<=i; ++j)
                              p[i].res+=p[j-1].bt;
                        p[i].res=p[i].res-p[i].at;
                  }
            }

            void show_data(Process pr[], int x)
            {
                  cout<<"\n--------------------------------------------------
--------------------------------";
                  cout<<"\nPid  ArrivalTime  BurstTime  WaitingTime
TurnaroundTime  ResponseTime\n";
                  cout<<"\n--------------------------------------------------
--------------------------------\n";
                  for(int c=1; c<=x; ++c)

      cout<<c<<"\t"<<pr[c].at<<"\t\t"<<pr[c].bt<<"\t"<<pr[c].wt<<"\t\t"<<pr[c
].tar<<"\t\t"<<pr[c].res<<endl;
                  cout<<"\nAverage Waiting Time: "<<avgwt;
                  cout<<"\nAverage Turnaround Time: "<<avgtt;
            }

};

int main()
{
      Process pro[10];
      int n;
      cout<<"\nEnter no. of processes (max 10) : ";
      cin>>n;
      do
      {
            if(n<0||n>10)
                  {
                        cout<<"\nEnter again : ";
                        cin>>n;
                  }
      }while(n<0||n>10);

      for(int i=1; i<=n; ++i)
      {
            cout<<"\nEnter details for "<<i<<" process: \n";
            pro[i].entry();
      }
      pro[1].sorter(pro, n);
      pro[1].cal_wait(pro, n);
      pro[1].cal_trn(pro, n);
      pro[1].cal_res(pro, n);
      pro[1].show_data(pro, n);

      return 0;
}
```

OUTPUT



```
Enter no. of processes (max 10) : 3

Enter details for 1 process:

Enter arrival time : 0

Enter burst time : 24

Enter details for 2 process:

Enter arrival time : 0

Enter burst time : 3

Enter details for 3 process:

Enter arrival time :
0

Enter burst time : 3

--------------------------------------------------------------------------------
--
Pid  ArrivalTime  BurstTime  WaitingTime  TurnaroundTime  ResponseTime

--------------------------------------------------------------------------------
--
1       0               24      0               24              0
2       0               3       24              27              24
3       0               3       27              30              27

Average Waiting Time: 17
Average Turnaround Time: 27
--------------------------------
Process exited after 17.78 seconds with return value 0
```

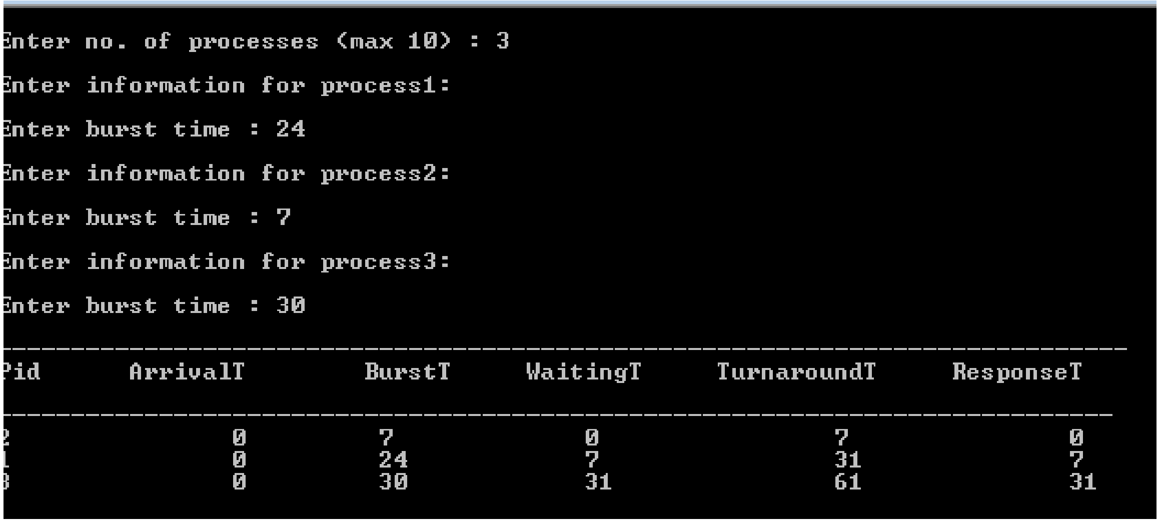
8. Write a program to implement SJF scheduling algorithm.

A8.

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

class Process
{
      int at;
      int bt;
      int wt;
      int tar;
      int res;
      int pid;

      public:
            void entry(int i)
            {
                  pid=i+1;
                  at=0;
                  cout<<"\nEnter burst time : ";
                  cin>>bt;

            }

            void sort(Process p[],int size)
            {
                  for(int i=0; i<size; ++i)
                  {
                        for(int j=0; j<size-1; ++j)
                        {
                              if(p[j].bt>p[j+1].bt)
                              {
                                    Process temp=p[j];
                                    p[j]=p[j+1];
                                    p[j+1]=temp;
                              }
                        }
                  }
            }

            void cal_wait(Process p[], int sz)
            {
                  p[0].wt=0;
                  for(int i=1; i<sz; )
                  {
                        p[i].wt=0;
                        for(int x=0; x<i; ++x)
                        {
                              p[i].wt+=p[x].bt;
                        }
                        p[i].wt-=p[i].at;

                        int j=i+1;
                        if(p[j].at<=p[i].bt)
                        {
                              i=j;
                        }
                        else
                        {
                              i++;
                        }
                  }
            }


            void cal_trn(Process p[], int s)
            {
                  for(int i=0; i<s; ++i)
                  p[i].tar=p[i].wt+p[i].bt;
            }


            void cal_res(Process p[], int y)
            {
                  p[0].res=0;
                  for(int i=1; i<y; ++i)
                  {
                        p[i].res=0;
                        for(int j=1; j<=i; ++j)
                              p[i].res+=p[j-1].bt;

                        p[i].res-=p[i].at;
                  }
            }


            void show_data(Process pr[], int x)
            {
                  cout<<"\n------------------------------------------------
--------------------------";
                  cout<<"\nPid\t ArrivalT\t BurstT\t    WaitingT\t 
TurnaroundT\t ResponseT\n";
                  cout<<"\n------------------------------------------------
-------------------------\n";
```

```
                  for(int c=0; c<x; ++c)
                  cout<<pr[c].pid<<"\t \t"<<pr[c].at<<"\t
"<<pr[c].bt<<"\t\t"<<pr[c].wt<<"\t \t "<<pr[c].tar<<"\t \t
"<<pr[c].res<<endl;
            }

};

int main()
{
      Process pro[10];
      int n;
      cout<<"\nEnter no. of processes (max 10) : ";
      cin>>n;
      do
      {
            if(n<0||n>10)
                  {
                        cout<<"\nEnter again : ";
                        cin>>n;
                  }
      }while(n<0||n>10);

      for(int i=0; i<n; ++i)
      {
            cout<<"\nEnter information for process"<<i+1<<": \n";
            pro[i].entry(i);
      }

      pro[0].sort(pro, n);
      pro[0].cal_wait(pro, n);
      pro[0].cal_trn(pro, n);
      pro[0].cal_res(pro, n);
      pro[0].show_data(pro, n);

      return 0;
}
```

OUTPUT



```
Enter no. of processes (max 10) : 3

Enter information for process1:

Enter burst time : 24

Enter information for process2:

Enter burst time : 7

Enter information for process3:

Enter burst time : 30

-----------------------------------------------------------------------------
Pid      ArrivalT        BurstT     WaitingT     TurnaroundT      ResponseT

----------------------------------------------------------------------------
2               0          7              0               7               0
1               0          24             7               31              7
3               0          30             31              61              31
```