11. Write a program to implement SRJF scheduling algorithm.

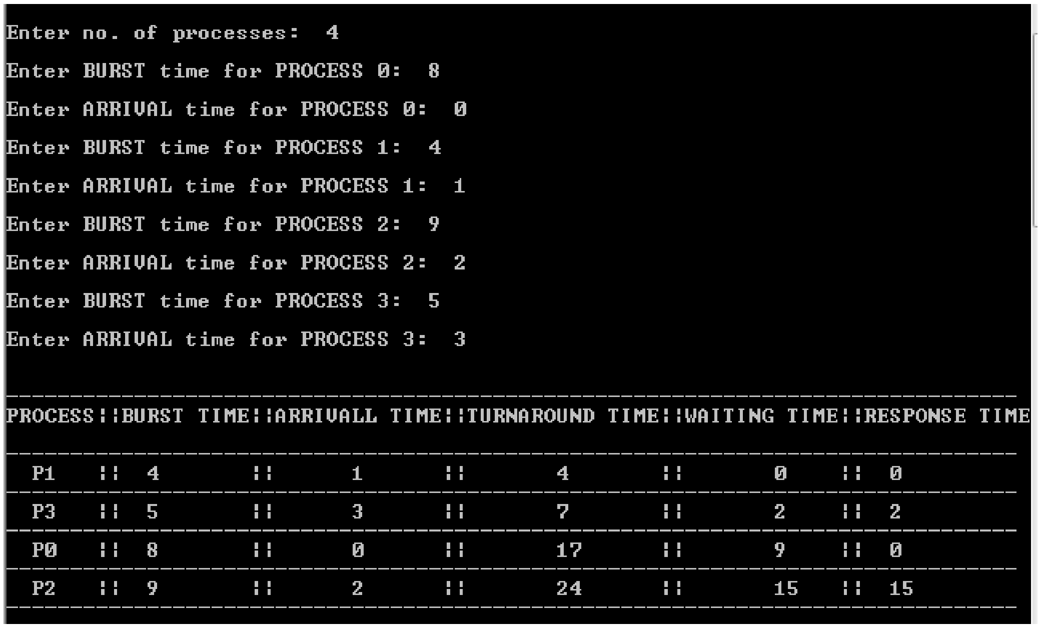
A.

CODE:

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

struct process
{
        int time;
        int burst;
        int turn;
        int arrival;
        int id;
        int response;
        int wait;
};

int main()
{

        int i,j,k,n,q,check=0;
        process p[10];
        process temp;
        cout<<"\nEnter no. of processes:  ";
        cin>>n;

        for(i=0;i<n;i++)
        {

                p[i].id=i;
                cout<<"\nEnter BURST time for PROCESS "<<i<<":  ";
                cin>>p[i].time;
                p[i].burst=p[i].time;
                cout<<"\nEnter ARRIVAL time for PROCESS "<<i<<":  ";
                cin>>p[i].arrival;
        p[i].turn=p[i].response=p[i].wait=0;
    }

    cout<<"\n";
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1-i;j++)
        {
            if(p[j].arrival>p[j+1].arrival)
            {
                temp=p[j];
                p[j]=p[j+1];
                p[j+1]=temp;
            }
        }
    }

process sm;
int min;
sm=p[0];
min=0;
for(i=0;i<n;i++)
{
    for(j=0;j<=i;j++)
    {
        if(p[j].time<sm.time)
        {
            sm=p[j];
            min=j;
        }
    }
    if(i<=n-2)
    {

    sm.time=sm.time+sm.arrival-p[i+1].arrival;
    p[min]=sm;
    }
    for(k=0;k<=i;k++)
    {
        if(k!=min && i<=n-2)
        {
            p[k].wait=p[k].wait-p[min].arrival+p[i+1].arrival;
        }
    }
}

p[n-1].wait=p[n-1].wait+1;


    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1-i;j++)
        {
            if(p[j].time>p[j+1].time)
            {
                temp=p[j];
                p[j]=p[j+1];
                p[j+1]=temp;
            }
        }
    }



for(i=0;i<n;i++)
{
    p[i].turn=p[i].burst+p[i].wait;
    for(j=0;j<i;j++)
    {
        p[i].turn=p[i].turn+p[j].time;
    }
    p[i].wait=p[i].turn-p[i].burst;
    if(p[i].arrival!=0 && p[i].wait!=0)
```

```
        p[i].response=p[i].wait;
}


        cout<<"\n------------------------------------------------------------------------------
";
        cout<<"\nPROCESS||BURST    TIME||ARRIVALL    TIME||TURNAROUND
TIME||WAITING TIME||RESPONSE TIME" ;
        cout<<"\n------------------------------------------------------------------------------
";

        for(i=0;i<n;i++)
        {
                cout<<"\n  P"<<p[i].id;
                cout<<"  ||  "<<p[i].burst;
                cout<<"\t   ||     "<<p[i].arrival;
                cout<<"     ||      "<<p[i].turn;
                cout<<"\t   ||      "<<p[i].wait;
                cout<<"\t || "<<p[i].response;
                cout<<"\n-----------------------------------------------------------------------
-------";



        }


}
```

OUTPUT:



```
Enter no. of processes:  4

Enter BURST time for PROCESS 0:  8

Enter ARRIVAL time for PROCESS 0:  0

Enter BURST time for PROCESS 1:  4

Enter ARRIVAL time for PROCESS 1:  1

Enter BURST time for PROCESS 2:  9

Enter ARRIVAL time for PROCESS 2:  2

Enter BURST time for PROCESS 3:  5

Enter ARRIVAL time for PROCESS 3:  3


-------------------------------------------------------------------------------
PROCESS||BURST TIME||ARRIVALL TIME||TURNAROUND TIME||WAITING TIME||RESPONSE TIME

-------------------------------------------------------------------------------
  P1   ||  4       ||      1      ||       4       ||       0    ||  0
-------------------------------------------------------------------------------
  P3   ||  5       ||      3      ||       7       ||       2    ||  2
-------------------------------------------------------------------------------
  P0   ||  8       ||      0      ||       17      ||       9    ||  0
-------------------------------------------------------------------------------
  P2   ||  9       ||      2      ||       24      ||       15   ||  15
-------------------------------------------------------------------------------
```

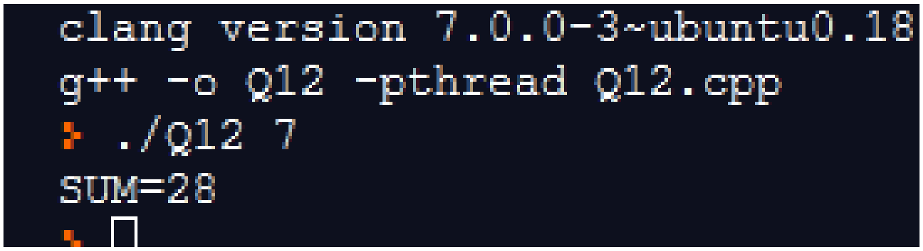
12. Write a program to calculate sum of n numbers using thread library.

A.

CODE:

```
#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
int sum;
void *runner(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, runner, argv[1]);
 pthread_join(tid, NULL);
 printf("SUM=%d\n", sum);
 return 0;
}
void *runner(void *param)
{
 int i, upper=atoi((char*)param);
 sum=0;
 for(i=1;i<=upper;i++)
   sum+=i;
 pthread_exit(0);
}
```

OUTPUT:



```
clang version 7.0.0-3~ubuntu0.18
g++ -o Q12 -pthread Q12.cpp
› ./Q12 7
SUM=28
›
```