Process Scheduling Algorithms in C++:

(SJF, RR, Priority Scheduling)

Code:

```
#include<iostream>

#include<string.h>

#include <conio.h>

#include <iomanip>

using namespace std;

struct Node{

        string process_name;

        int brust_time;

        int priority;

        Node *next;

        Node(string p_n,int b,int p){

                process_name = p_n;

                brust_time = b;

                priority=p;

                next =NULL;

        }

};

class Queue{

        Node *rear;

        Node *front;

        int size;

        void line(){
```

```
        for(int a=1;a<=100;a++){
                cout <<"_";}cout<<endl;}
void line(int n){
        for(int a=1;a<=n;a++){
                cout <<"_";}cout<<endl;}
string process_name_value(){
        string name; cout<< "Enter the Process Name  : ";getline(cin,name);
        return name; }
int brust_time_value(){
        int v; cout<< "Enter the brust time    : ";cin>>v;cin.ignore();
        return v; }
int priority_value(){
        int v; cout<< "Enter the priority      : ";cin>>v;cin.ignore();
        return v; }
Node* swap(Node* ptr1, Node* ptr2){
struct Node* tmp = ptr2->next;ptr2->next = ptr1;ptr1->next = tmp;return ptr2;}
void bubbleSort_Brust_time(Node **head){
        Node** h;int i, j, swapped;
        for (i = 0; i <= size; i++){
        h = head;swapped = 0;
        for (j = 0; j < size - i - 1; j++){
                Node* p1 = *h;Node* p2 = p1->next;
                if (p1->brust_time > p2->brust_time){
                *h = swap(p1, p2);swapped = 1;}h = &(*h)->next;}
        if (swapped == 0)  break;}              }
```

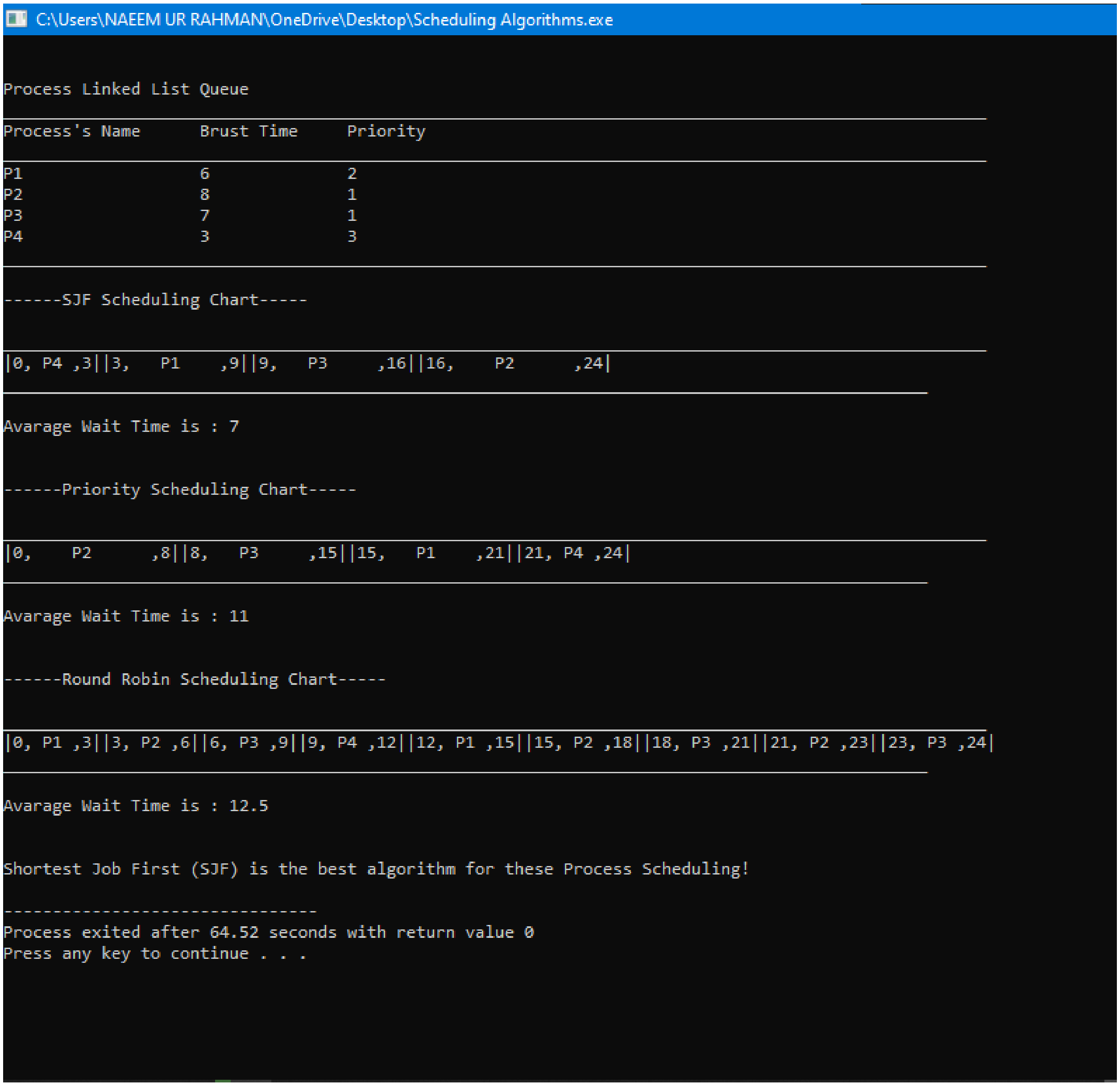
```
void bubbleSort_Priority(Node **head){
    Node** h; int i, j, swapped;
    for (i = 0; i <= size; i++){
    h = head;swapped = 0;
    for (j = 0; j < size - i - 1; j++){
        Node* p1 = *h; Node* p2 = p1->next;
        if (p1->priority > p2->priority){
            /* update the link after swapping */
        *h = swap(p1, p2); swapped = 1;} h = &(*h)->next;}
            if (swapped == 0)
    break;}               }
int orignal_brust_time(string name,Queue q){
    Node* temp = q.front;
    while(temp!=NULL){
    if(temp->process_name==name){
        return temp->brust_time;}
    temp = temp->next;}}
public :
    Queue(){
        front = NULL;
        rear = NULL;
        size = 0;}
    void enqueue(){
        if(front==NULL){
front = new Node(process_name_value(),brust_time_value(),priority_value());
```

```
            size++;rear =  front;
            }else{Node *temp = new Node(process_name_value(),brust_time_value(),priority_value());
              size++;rear->next = temp;rear = temp;
            }}
        void dequeue(){
            if(front==NULL){
                cout<< "Under Flow Queue"<<endl;
            }else{
                //cout << "Dequeue Value is : "<<front->process_name<<endl;
                front = front->next;
                size--;
                if(front == NULL){
                    rear = NULL;}}}
        void peek(){
            if (front==NULL){
                cout << "Empty Queue "<<endl;
            }else{
                cout<< "Peek Value is : "<<front->process_name<<endl;}}
        void display(){
            if(front==NULL){
                cout <<"Empty Queue"<<endl;return ;}
            cout << "Process Linked List Queue"<<endl;
            Node *temp = front;
                line();
```

```
cout<<setiosflags(ios::left)<<setw(20)<<"Process's
Name"<<setiosflags(ios::left)<<setw(15)<<"Brust
Time"<<setiosflags(ios::left)<<setw(15)<<"Priority"<<endl; line();

            while(temp!=NULL){

cout <<setiosflags(ios::left)<<setw(20)<<temp-
>process_name<<setiosflags(ios::left)<<setw(15)<<temp-
>brust_time<<setiosflags(ios::left)<<setw(15)<<temp->priority<<endl;

                temp = temp->next;}

            line();}

        void copy_linkded_list(Queue q){

        Node* tempf = q.front;

        while(tempf!=NULL){

        if(front==NULL){

    front = new Node(tempf->process_name,tempf->brust_time,tempf->priority);

        size++;rear =  front;

            }else{

Node *temp = new Node(tempf->process_name,tempf->brust_time,tempf->priority);

    size++;rear->next = temp;rear = temp;}tempf= tempf->next;}}

        float SJF(){

            if(front==NULL){

                cout <<"Empty Queue"<<endl;return 0 ;}

            bubbleSort_Brust_time(&front);

            int t = 0,newt=0;

            int avg_wait_time=0,no_of_process=0;

            Node *temp = front;cout<<endl;

            cout<<"------SJF Scheduling Chart-----"<<endl<<endl;line();

            while(temp!=NULL){
                avg_wait_time+=t;

                newt += temp->brust_time;

cout <<"|"<<t<<","<<setiosflags(ios::left)<<setw(temp-
>brust_time/2)<<""<<setiosflags(ios::left)<<setw(temp->brust_time)<<temp-
>process_name <<","<<newt<<"|";

              t=newt;temp = temp->next;no_of_process++;   }

            cout<<endl;line(newt+70);cout<<endl;

            cout<<"Avarage Wait Time is :
"<<(float)avg_wait_time/no_of_process<<endl<<endl;

            return (float)avg_wait_time/no_of_process;}

        float Priority(){

            if(front==NULL){

                cout <<"Empty Queue"<<endl;

                return 0 ;}

            bubbleSort_Priority(&front);

            int t = 0,newt=0;

            int avg_wait_time=0,no_of_process=0;

            Node *temp = front;

            cout<<endl;cout<<"------Priority Scheduling Chart-----
"<<endl<<endl;line();

            while(temp!=NULL || size==0){

                avg_wait_time+=t;

                newt += temp->brust_time;

                cout <<"|"<<t<<","<<setiosflags(ios::left)<<setw(temp-
>brust_time/2)<<""

                      <<setiosflags(ios::left)<<setw(temp-
>brust_time)<<temp->process_name
```

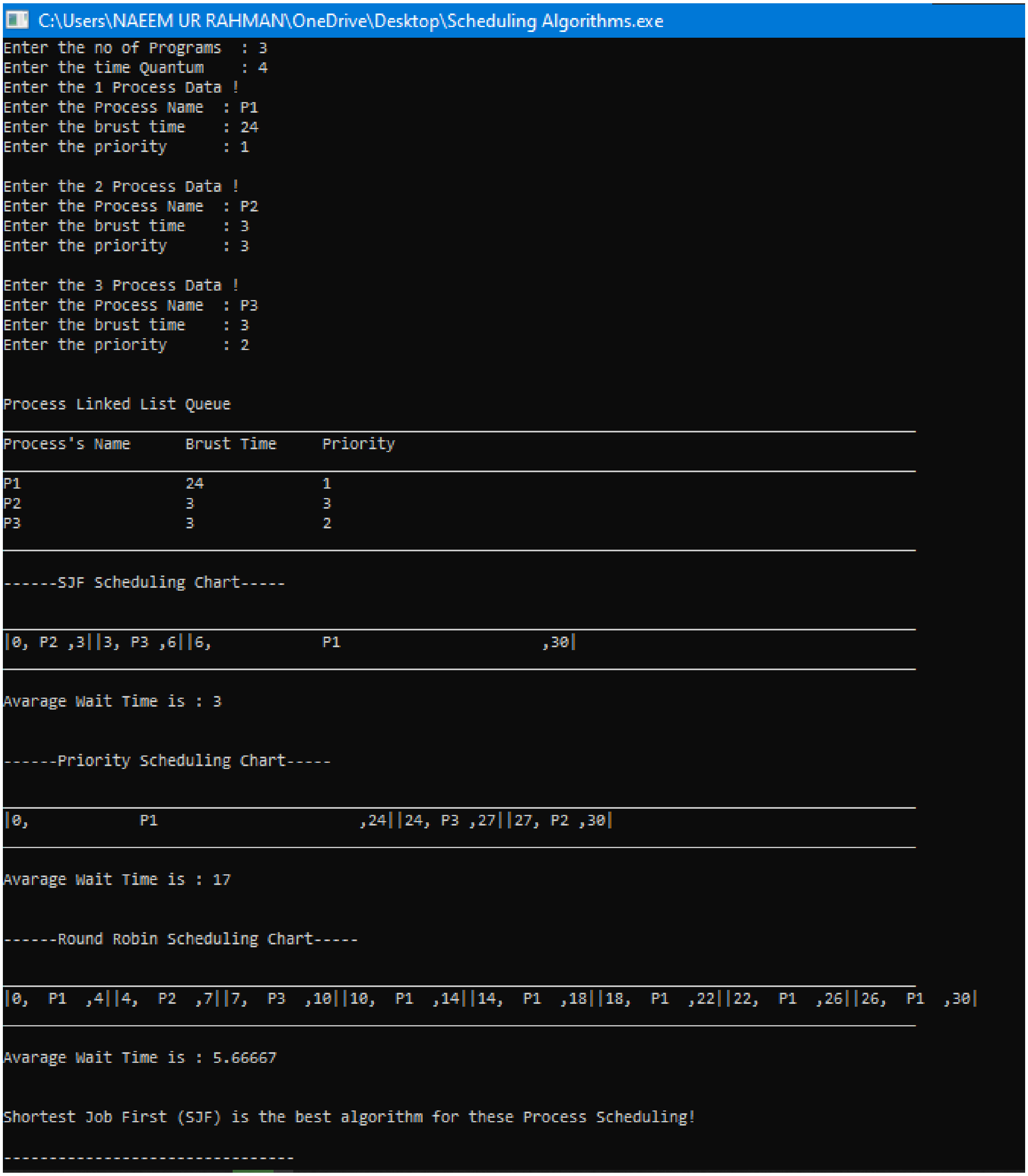
```
                                                <<","<<newt<<"|";

                              t=newt;temp = temp->next;no_of_process++;    }

                            cout<<endl;line(newt+70);cout<<endl;

                            cout<<"Avarage Wait Time is :
"<<(float)avg_wait_time/no_of_process<<endl<<endl;

                            return (float) avg_wait_time/no_of_process;}

                    float RR(int quantum,Queue q){

                            if(front==NULL){

                                    cout <<"Empty Queue"<<endl;return 0;}

                            int t = 0,newt=0,no_process=size;

                            int avg_wait_time=0;cout<<endl;

                    cout<<"------Round Robin Scheduling Chart-----"<<endl<<endl;line();

                            while(front!=NULL){

                                    if(front->brust_time<=quantum){

                                                newt += front->brust_time;

                     cout <<"|"<<t<<","<<setiosflags(ios::left)<<setw(quantum/2)<<""

                                                    <<setiosflags(ios::left)<<setw(quantum)<<front-
>process_name <<","<<newt<<"|";

                                                avg_wait_time += newt - orignal_brust_time(front-
>process_name,q);

                                                t=newt; dequeue();

                                    }else{

                                               newt += quantum;

                    cout <<"|"<<t<<","<<setiosflags(ios::left)<<setw(quantum/2)<<""

                                                    <<setiosflags(ios::left)<<setw(quantum)<<front-
>process_name <<","<<newt<<"|";

                                                t=newt;
                    string name = front->process_name;
                    int b_t = front->brust_time-quantum;
                    int p = front->priority; dequeue();
                    if(front==NULL){
                        front = new Node(name,b_t,p);
                        size++;
                        rear =  front;
                    }else{
                        Node *temp = new Node(name,b_t,p);
                        size++;
                        rear->next = temp;
                        rear = temp;
                    }}}
            cout<<endl;line(newt+70);cout<<endl;
    cout<<"Avarage Wait Time is : "<<(float)avg_wait_time/no_process<<endl<<endl;
            return (float) avg_wait_time/no_process;
        }};
int main(){
    int n,quantum;
    cout<<"Enter the no of Programs  : ";cin>>n;cin.ignore();
    cout<<"Enter the time Quantum    : ";cin>>quantum;cin.ignore();
    Queue q ;
    for (int i=1;i<=n;i++){
        cout<<"Enter the "<<i<<" Process Data ! "<<endl;
        q.enqueue();cout<<endl;}
    cout<<endl;q.display();

    Queue q1 , q2, q3;

    q1.copy_linkded_list(q);q2.copy_linkded_list(q);q3.copy_linkded_list(q);

    float SJF = q1.SJF();

    float Pri = q2.Priority();

    float RR  = q3.RR(quantum,q);

    cout<<endl;

    if(SJF<Pri){

    if(SJF<RR){

cout<<"Shortest Job First (SJF) is the best algorithm for these Process Scheduling! "<<endl;

        }else{

cout<<"Round Robin (RR) is the best algorithm for these Process Scheduling! "<<endl;}

    }else{

        if(Pri<RR){

cout<<"Priority Scheduling is the best algorithm for these Process Scheduling! "<<endl;

        }else{

cout<<"Round Robin (RR) is the best algorithm for these Process Scheduling! "<<endl;

        }        }

    return 0;        }
```

Output Example 1:


C:\Users\NAEEM UR RAHMAN\OneDrive\Desktop\Scheduling Algorithms.exe

```
Process Linked List Queue

Process's Name      Brust Time     Priority

P1                  6              2
P2                  8              1
P3                  7              1
P4                  3              3

------SJF Scheduling Chart-----

|0, P4 ,3||3,    P1    ,9||9,    P3     ,16||16,    P2      ,24|

Avarage Wait Time is : 7

------Priority Scheduling Chart-----

|0,    P2      ,8||8,    P3     ,15||15,    P1     ,21||21, P4 ,24|

Avarage Wait Time is : 11

------Round Robin Scheduling Chart-----

|0, P1 ,3||3, P2 ,6||6, P3 ,9||9, P4 ,12||12, P1 ,15||15, P2 ,18||18, P3 ,21||21, P2 ,23||23, P3 ,24|

Avarage Wait Time is : 12.5

Shortest Job First (SJF) is the best algorithm for these Process Scheduling!

--------------------------------
Process exited after 64.52 seconds with return value 0
Press any key to continue . . .
```

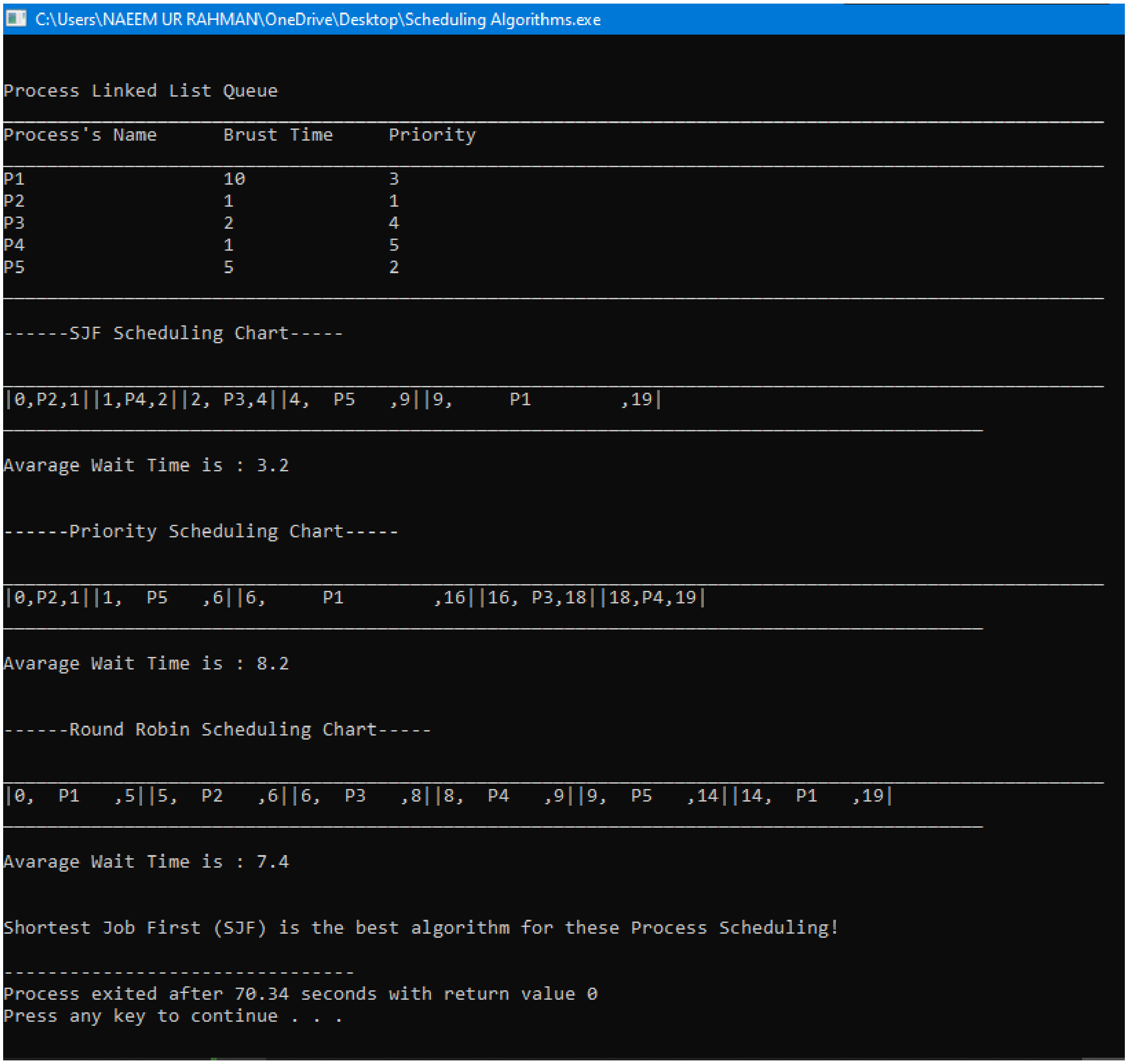
Output Example 2:



C:\Users\NAEEM UR RAHMAN\OneDrive\Desktop\Scheduling Algorithms.exe

```
Enter the no of Programs  : 3
Enter the time Quantum    : 4
Enter the 1 Process Data !
Enter the Process Name  : P1
Enter the brust time    : 24
Enter the priority      : 1

Enter the 2 Process Data !
Enter the Process Name  : P2
Enter the brust time    : 3
Enter the priority      : 3

Enter the 3 Process Data !
Enter the Process Name  : P3
Enter the brust time    : 3
Enter the priority      : 2


Process Linked List Queue
____________________________________________________________________________________________________
Process's Name      Brust Time     Priority
____________________________________________________________________________________________________
P1                  24             1
P2                  3              3
P3                  3              2
____________________________________________________________________________________________________

------SJF Scheduling Chart-----

____________________________________________________________________________________________________
|0, P2 ,3||3, P3 ,6||6,            P1                      ,30|
____________________________________________________________________________________________________

Avarage Wait Time is : 3

------Priority Scheduling Chart-----

____________________________________________________________________________________________________
|0,            P1                      ,24||24, P3 ,27||27, P2 ,30|
____________________________________________________________________________________________________

Avarage Wait Time is : 17

------Round Robin Scheduling Chart-----

____________________________________________________________________________________________________
|0,  P1  ,4||4,  P2  ,7||7,  P3  ,10||10,  P1  ,14||14,  P1  ,18||18,  P1  ,22||22,  P1  ,26||26,  P1  ,30|
____________________________________________________________________________________________________

Avarage Wait Time is : 5.66667

Shortest Job First (SJF) is the best algorithm for these Process Scheduling!

--------------------------------
```

Output Example 3:



C:\Users\NAEEM UR RAHMAN\OneDrive\Desktop\Scheduling Algorithms.exe

```

Process Linked List Queue
____________________________________________________________________________________________________
Process's Name      Brust Time     Priority
____________________________________________________________________________________________________
P1                  10             3
P2                  1              1
P3                  2              4
P4                  1              5
P5                  5              2
____________________________________________________________________________________________________

------SJF Scheduling Chart-----

____________________________________________________________________________________________________
|0,P2,1||1,P4,2||2, P3,4||4,  P5   ,9||9,     P1         ,19|
_________________________________________________________________________________________

Avarage Wait Time is : 3.2


------Priority Scheduling Chart-----

____________________________________________________________________________________________________
|0,P2,1||1,  P5   ,6||6,     P1         ,16||16, P3,18||18,P4,19|
_________________________________________________________________________________________

Avarage Wait Time is : 8.2


------Round Robin Scheduling Chart-----

____________________________________________________________________________________________________
|0,  P1   ,5||5,  P2   ,6||6,  P3   ,8||8,  P4   ,9||9,  P5   ,14||14,  P1   ,19|
_________________________________________________________________________________________

Avarage Wait Time is : 7.4


Shortest Job First (SJF) is the best algorithm for these Process Scheduling!

--------------------------------
Process exited after 70.34 seconds with return value 0
Press any key to continue . . .
```