**20BCS042 | MOHD ADIL**

**PROGRAM:3 -> SHORTEST JOB FIRST NON-PREEMPTIVE SCHEDULING ALGORITHM**

---

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

struct Process{
    char Pname[3];
    int arvlTime;
    int brstTime;
    int cmplTime;
    int wtngTime;
    int tartTime;
    int respTime;
};

struct priorityQ{
    Process pr;
    priorityQ *next;
};

priorityQ *push(priorityQ *front, Process pSample, char b){
    priorityQ *node = new priorityQ;
    node->pr = pSample;
    node->next=NULL;

//means push according to burst Time
    if(b=='b'){
        if(front==NULL){
            front=node;
        }
        else if(front->pr.brstTime > pSample.brstTime){
            node->next=front;
            front=node;
        }
        else{
            priorityQ *tmp=front;
            while (tmp->next!=NULL && tmp->next->pr.brstTime <
pSample.brstTime){
                tmp=tmp->next;
```

```
            }

            node->next=tmp->next;
            tmp->next=node;
        }
    }

//otherwise push accoring to arrival time
    else{
        if(front==NULL){
            front=node;
        }
        else if(front->pr.arvlTime > pSample.arvlTime){
            node->next=front;
            front=node;
        }
        else{
            priorityQ *tmp=front;
            while (tmp->next!=NULL && tmp->next->pr.arvlTime <
pSample.arvlTime){
                tmp=tmp->next;
            }

            node->next=tmp->next;
            tmp->next=node;
        }
    }

    return front;
}

priorityQ *pop(priorityQ *front){
    front=front->next;
    return front;
}

Process top(priorityQ *front){
    return front->pr;
}

bool empty(priorityQ *front){
    return (front==NULL);
```

```
}

//ans vector
vector<Process> v;
int n;
float avgc, avgw, avgt;

void SJF(priorityQ *pq1){
    int cmpt = top(pq1).brstTime;
    v.push_back(top(pq1));
    pq1 = pop(pq1);

    priorityQ *pq2=NULL;

    while(!empty(pq1)){
        while(!empty(pq1) && top(pq1).arvlTime < cmpt){
            pq2 = push(pq2, top(pq1), 'b');
            pq1 = pop(pq1);
        }

        cmpt += top(pq2).brstTime;
        v.push_back(top(pq2));
        pq2 = pop(pq2);
    }

    while(!empty(pq2)){
        v.push_back(top(pq2));
        pq2 = pop(pq2);
    }
}

void calculateTimes(){
    v.front().wtngTime=0;
    v.front().cmplTime = v.front().brstTime;
    float sumc=0, sumw=0, sumt=0;

    //calculating completion time
    int prv = v.front().cmplTime;
    sumc += prv;
    for(int i=1; i<n; i++){
        v[i].cmplTime = prv + v[i].brstTime;
        prv = v[i].cmplTime;
```

```
        sumc += v[i].cmplTime;
    }

    //calculating waiting time
    prv = v.front().cmplTime;
    for(int i=1; i<n; i++){
        v[i].wtngTime = prv - v[i].arvlTime;
        prv = v[i].cmplTime;
        sumw += v[i].wtngTime;
    }

    //calculating turn around time
    for(int i=0; i<n; i++){
        v[i].tartTime = v[i].brstTime + v[i].wtngTime;
        sumt += v[i].tartTime;
    }

    //calculating avg(s) time
    avgc = sumc/n;
    avgw = sumw/n;
    avgt = sumt/n;
}

void display(){
    cout<<"\n\nDisplaying the table :- ";

    cout<<"\n\n+--------------+------------+--------------+-----------
------+--------------+-----------------+---------------+";
    cout<<"\n| Process name | Burst Time | Arrival Time | Completion
Time | Waiting Time | TurnAround Time | Response Time |";
    cout<<"\n+--------------+------------+--------------+------------
----+--------------+-----------------+---------------+";

    for(auto i:v){
        printf("\n|      %s      |    %2d      |      %2d      |
 %2d        |      %2d      |      %2d          |      %2d       |"
               ,i.Pname, i.brstTime, i.arvlTime, i.cmplTime,
i.wtngTime, i.tartTime, i.wtngTime);
    cout<<"\n+--------------+------------+--------------+------------
----+--------------+-----------------+---------------+";
    }
```

```
    cout<<"\n\n";
    printf("\nAverage Completion time : %.2fns", avgc);
    printf("\nAverage Waiting time : %.2fns", avgw);
    printf("\nAverage TurnAround time : %.2fns", avgt);
    printf("\nAverage Response time : %.2fns", avgw);
}

void printGantt(){

    cout<<"\n\nGantt Chart : ";

    cout<<"\n\n+";
    for(auto p:v){
        for(int i=0; i<2*p.brstTime; i++){
            cout<<"-";
        }
        cout<<"+";
    }

    cout<<"\n|";
    for(auto p:v){
        for(int i=0; i<p.brstTime-1; i++){
            cout<<" ";
        }
        cout<<p.Pname;
        for(int i=0; i<p.brstTime-1; i++){
            cout<<" ";
        }
        cout<<"|";
    }

    cout<<"\n+";
    for(auto p:v){
        for(int i=0; i<2*p.brstTime; i++){
            cout<<"-";
        }
        cout<<"+";
    }

    cout<<"\n0";
    for(auto p:v){
        for(int i=0; i<2*p.brstTime-1; i++){
```

```
            cout<<" ";
        }
        printf("%2d", p.cmplTime);
    }
    cout<<"\n\n";
}

int main(){
    priorityQ *pq1=NULL;

    cout<<"Enter the no of the Processes : ";
    cin>>n;

    for(int i=0; i<n; i++){
        struct Process p;
        cout<<"Enter Process "<<i+1<<" name, its burst Time and
Arrival Time : ";
        cin>>p.Pname>>p.brstTime>>p.arvlTime;
        pq1 = push(pq1, p, 'a');
    }

    //sort according to arrival time + burst Time :
    SJF(pq1);
    calculateTimes();


    display();
    printGantt();
    return 0;
}
```

## OUTPUT:

```
Enter the no of the Processes : 5
Enter Process 1 name, its burst Time and Arrival Time : P1 6 2
Enter Process 2 name, its burst Time and Arrival Time : P2 2 5
Enter Process 3 name, its burst Time and Arrival Time : P3 8 1
Enter Process 4 name, its burst Time and Arrival Time : P4 3 0
Enter Process 5 name, its burst Time and Arrival Time : P5 4 4


Displaying the table :-

+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+
| Process name | Burst Time | Arrival Time | Completion Time | Waiting Time | TurnAround Time | Response Time |
+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+
|      P4      |     3      |       0      |         3       |       0      |       3         |       0       |
+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+
|      P1      |     6      |       2      |         9       |       1      |       7         |       1       |
+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+
|      P2      |     2      |       5      |        11       |       4      |       6         |       4       |
+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+
|      P5      |     4      |       4      |        15       |       7      |      11         |       7       |
+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+
|      P3      |     8      |       1      |        23       |      14      |      22         |      14       |
+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+


Average Completion time : 12.20ns
Average Waiting time : 5.20ns
Average TurnAround time : 9.80ns
Average Response time : 5.20ns

Gantt Chart :

+------+------------+----+--------+----------------+
|  P4  |     P1     | P2 |   P5   |       P3       |
+------+------------+----+--------+----------------+
0      3            9   11       15               23
```