Name: Bilal Alakhail

Reg: 11815035

Email: alakhail.bilal@gmail.com

Github Link:

Question: Consider a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times. The priority can be computed as :
Priority = 1+ Waiting time / Estimated run time
Using the data given below compute the waiting time and turnaround time for each process and average waiting time and average turnaround time.

Description:

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

- Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.
- It is a Greedy Algorithm.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.
- It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

Algorithm:

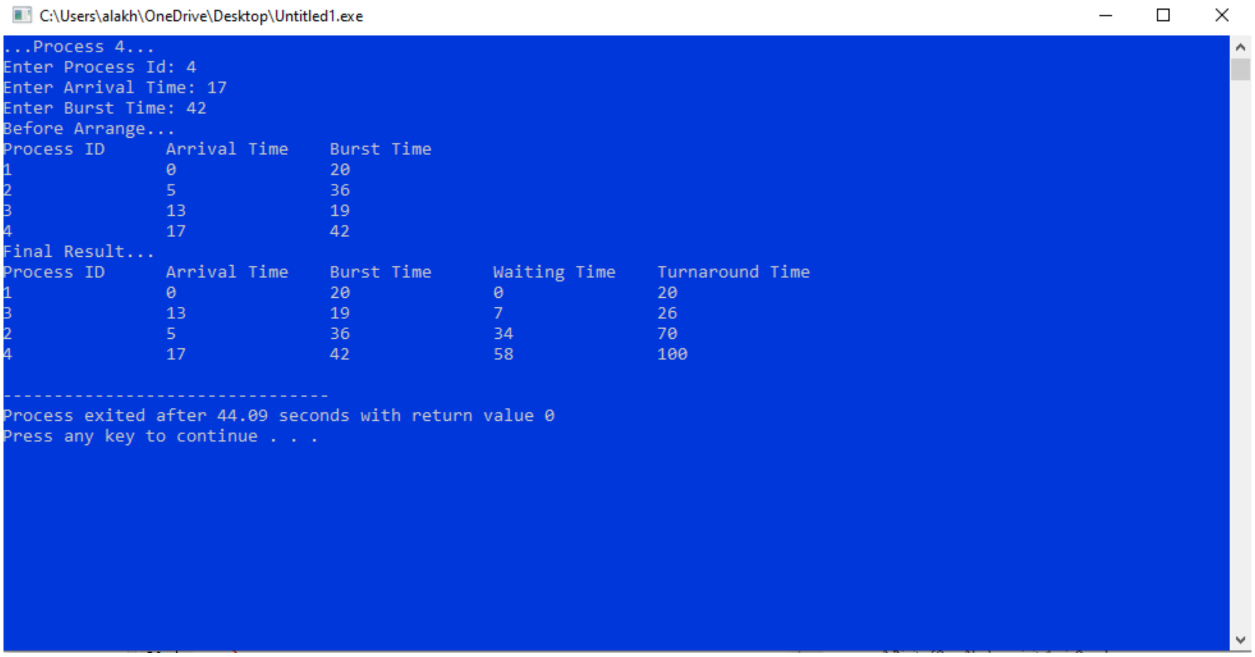
1. Sort all the process according to the arrival time.
2. Then select that process which has minimum arrival time and minimum Burst time.
3. After completion of process make a pool of process which after till the completion of previous process and select that process among the pool which is having minimum Burst time.

How to compute below times in SJF using a program?

1. Completion Time: Time at which process completes its execution.
2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time(W.T): Time Difference between turn around time and burst time. Waiting Time = Turn Around Time – Burst Time

Code:

```
#include<iostream>
using namespace std;
int mat[10][6];

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void arrangeArrival(int num, int mat[][6])
{
    for(int i=0; i<num; i++)
    {
        for(int j=0; j<num-i-1; j++)
        {
            if(mat[j][1] > mat[j+1][1])
            {
                for(int k=0; k<5; k++)
                {
                    swap(mat[j][k], mat[j+1][k]);
                }
            }
        }
    }
}

void completionTime(int num, int mat[][6])
{
    int temp, val;
    mat[0][3] = mat[0][1] + mat[0][2];
    mat[0][5] = mat[0][3] - mat[0][1];
    mat[0][4] = mat[0][5] - mat[0][2];

    for(int i=1; i<num; i++)
    {
        temp = mat[i-1][3];
        int low = mat[i][2];
        for(int j=i; j<num; j++)
        {
            if(temp >= mat[j][1] && low >= mat[j][2])
            {
                low = mat[j][2];
                val = j;
            }
        }
        mat[val][3] = temp + mat[val][2];
        mat[val][5] = mat[val][3] - mat[val][1];
        mat[val][4] = mat[val][5] - mat[val][2];
        for(int k=0; k<6; k++)
        {
            swap(mat[val][k], mat[i][k]);
        }
    }
}

int main()
{
    int num, temp;

    cout<<"Enter number of Process: ";
    cin>>num;

    cout<<"...Enter the process ID...\n";
    for(int i=0; i<num; i++)
    {
        cout<<"...Process "<<i+1<<"...\n";
        cout<<"Enter Process Id: ";
        cin>>mat[i][0];
        cout<<"Enter Arrival Time: ";
        cin>>mat[i][1];
        cout<<"Enter Burst Time: ";
        cin>>mat[i][2];
    }

    cout<<"Before Arrange...\n";
    cout<<"Process ID\tArrival Time\tBurst Time\n";
    for(int i=0; i<num; i++)
    {
        cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i][2]<<"\n";
    }

    arrangeArrival(num, mat);
    completionTime(num, mat);
    cout<<"Final Result...\n";
    cout<<"Process  ID\tArrival  Time\tBurst  Time\tWaiting  Time\tTurnaround
Time\n";
    for(int i=0; i<num; i++)
    {
        cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i][2]<<"\t\t"<<mat[i]
[4]<<"\t\t"<<mat[i][5]<<"\n";
    }
}
```



```
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 17
Enter Burst Time: 42
Before Arrange...
Process ID      Arrival Time    Burst Time
1               0               20
2               5               36
3               13              19
4               17              42
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               0               20              0               20
3               13              19              7               26
2               5               36              34              70
4               17              42              58              100

--------------------------------
Process exited after 44.09 seconds with return value 0
Press any key to continue . . .
```