Assignment of Operating System Lab

Topic: SJF Preemptive Scheduling

Name: Md. Farhan Masud Shohag

ID: B190305043

Question 1: Implement the Shortest Job First (SJF) scheduling algorithm with preemption. Write a program that reads a set of processes and their arrival times and burst times and simulates the execution of these processes using the SJF algorithm with preemption. The program should also calculate and output the average waiting time, and turnaround time. You also have to print the Gant chart using structure and pointer of C language.

Solution in C++ code:

```
//Farhan Masud Shohag (B190305043)
#include<iostream>
#include<algorithm>
using namespace std;

int cnt1, cnt2, cnt3;

struct Process {
   int process;
   int burst;
   int arrival;
   int response = 0;
   int compile = 0;
   int waiting = 0;
} arr1[100], arr2[100], arr3[100];

bool sort_burst_time(Process a, Process b) {
   return a.burst < b.burst;
}
bool sort_arrival_time(Process a, Process b) {
   return a.arrival < b.arrival;
}
bool sort_reverse_burst_time(Process a, Process b) {
   if (a.burst != b.burst)
      return a.burst > b.burst;
   return a.arrival < b.arrival;
```

```
}

void calculate(int size, int queue) {
    cnt1=cnt2=cnt3=0;
    int n = size, q;
    sort(arr1, arr1 + n, sort_arrival_time);
    int total_time = 0, i;
    int j, totalArray[n];
    int temp = 0;
    bool last = false;
    for (i = 0; i < n; i++) {
        temp += arr1[i].burst;
    }
    temp += arr1[0].arrival;
    for (i = 0; total_time <= temp;) {
        j = i;
        while (arr1[j].arrival <= total_time && j != n) {
            arr2[cnt3] = arr1[j];
            j++;
            cnt3++;
        }
        if (cnt3 == cnt2) {
            arr3[cnt1].process = 'i';
            arr3[cnt1].burst = arr1[j].arrival - total_time;
            arr3[cnt1].arrival = total_time;
            total_time += arr3[cnt1].burst;
            cnt1++;
            continue;
        }
        i = j;
        if (last == true) {
            sort(arr2 + cnt2, arr2 + cnt3, sort_burst_time);
        }

        j = cnt2;
        if (arr2[j].burst > queue) {
            arr3[cnt1] = arr2[j];
            arr3[cnt1].burst = queue;
            cnt1++;
            arr2[j].burst = arr2[j].burst - queue;
            total_time += queue;
            last = true;
            for (q = 0; q < n; q++) {
                if (arr2[j].process != arr1[q].process) {
                    arr1[q].waiting += queue;
                }
            }
        }
        else {
            arr3[cnt1] = arr2[j];
            cnt1++;
            cnt2++;
            total_time += arr2[j].burst;
            last = false;
            for (q = 0; q < n; q++) {
                if (arr2[j].process != arr1[q].process) {
                    arr1[q].waiting += arr2[j].burst;
                }
            }
        }
        if (cnt2 == cnt3 && i >= n)
            break;
    }
    totalArray[i] = total_time;
    total_time += arr1[i].burst;
    for (i = 0; i < cnt1 - 1; i++) {
        if (arr3[i].process == arr3[i + 1].process) {
            arr3[i].burst += arr3[i + 1].burst;
            for (j = i + 1; j < cnt1 - 1; j++)
                arr3[j] = arr3[j + 1];
            cnt1--;
            i--;
        }
    }

    int response_time = 0;
    for (j = 0; j < n; j++) {
        response_time = 0;
        for (i = 0; i < cnt1; i++) {
            if (arr3[i].process == arr1[j].process) {
                arr1[j].response = response_time;
                break;
            }
            response_time += arr3[i].burst;
        }
    }
    float average_Waiting_Time = 0,average_Response_Time = 0,
average_Turn_Around_Time = 0;

    cout << "\nGantt Chart:\n";
    response_time = 0;
    for (i = 0; i < cnt1; i++) {
        if (i != cnt1)
            cout << "|  " << 'P' << arr3[i].process << "   ";
        response_time += arr3[i].burst;
        for (j = 0; j < n; j++) {
            if (arr1[j].process == arr3[i].process)
                arr1[j].compile = response_time;
        }
    }
    cout << "|\n";
    response_time = 0;
    for (i = 0; i < cnt1 + 1; i++) {
        cout << response_time << "\t";
        totalArray[i] = response_time;
        response_time += arr3[i].burst;
    }

    cout << "\n\n";
    cout<<"--------------------------------------------------------------------------------\n";
    cout << "|    Process    |   Arrival T.  |    Burst T.   | TurnAround T. |   Waiting T.  |\n";
    cout<<"--------------------------------------------------------------------------------\n";
    for (i = 0; i < size && arr1[i].process != 'i'; i++) {
        if (arr1[i].process == '\0')
            break;
        cout << "|\tP" << arr1[i].process << "\t|";
        cout <<"\t"<< arr1[i].arrival << "\t|";
        cout <<"\t"<< arr1[i].burst << "\t|";
        cout <<"\t"<<arr1[i].waiting + arr1[i].compile - response_time + arr1[i].burst << "\t|";
        average_Turn_Around_Time += arr1[i].waiting + arr1[i].compile - response_time +
arr1[i].burst;
        cout <<"\t"<<arr1[i].waiting + arr1[i].compile - response_time << "\t|";
        average_Waiting_Time += arr1[i].waiting + arr1[i].compile - response_time;
        cout <<"\n";
    }
    cout<<"--------------------------------------------------------------------------------\n";

    cout << "\n\n";
    cout << "Average Waiting time: " << (float)average_Waiting_Time / (float)n << endl;
    cout << "Average Turn Around time: " << (float)average_Turn_Around_Time / (float)n <<
endl;

}

int main() {
    int n, queue;
    cout << "Enter number of Processes: ";
    cin >> n;
    cout << "Enter Arrival Time, Burst Time:\n";
    for (int i = 0; i < n; i++) {
        cout << "\tP" << i + 1 << ": ";
        arr1[i].process = i + 1;
        cin >> arr1[i].arrival;
        cin >> arr1[i].burst;
        arr1[i].waiting = -arr1[i].arrival + 1;
    }
    calculate(n, 1);
    return 0;
}
```