OS Lab Assignment 3

Submitted By:

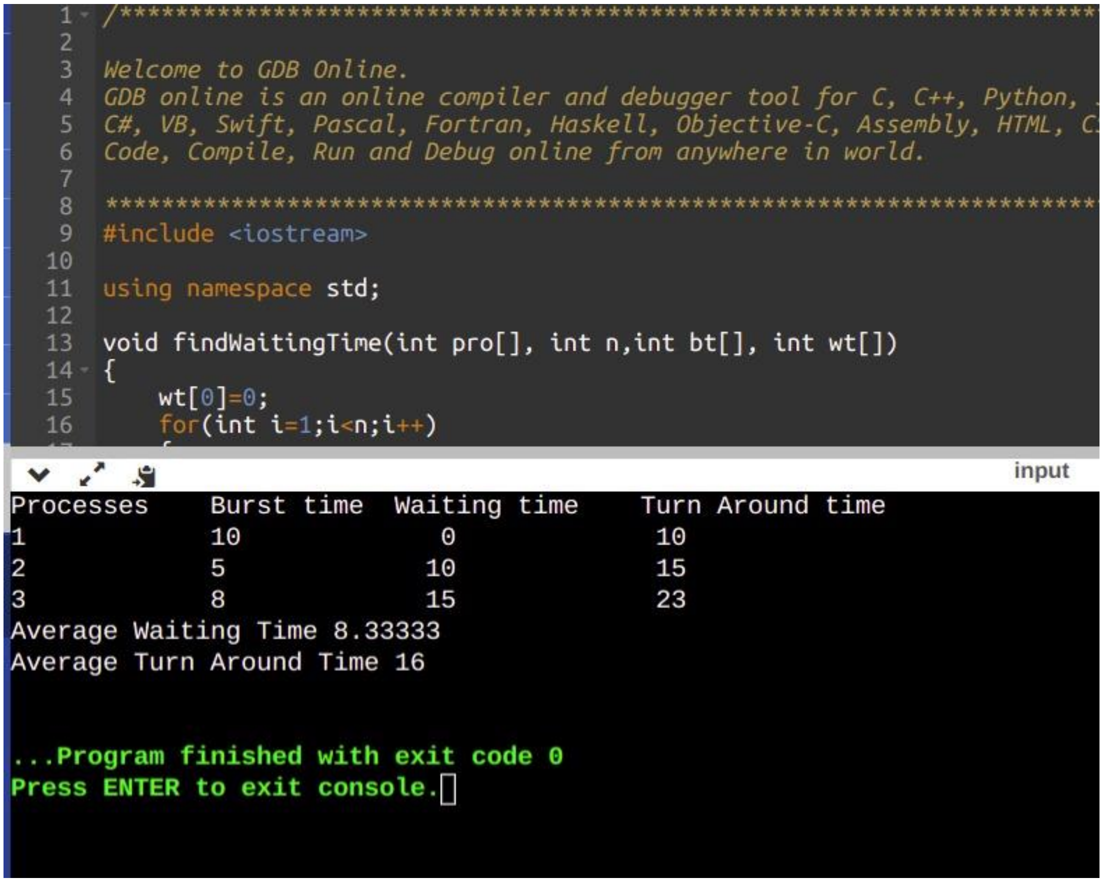
Manroop Parmar

101906134 3EC6

1. FCFS Algorithm

```
#include <iostream>

using namespace std;

void findWaitingTime(int pro[], int n,int bt[], int wt[])
{
   wt[0]=0;
   for(int i=1;i<n;i++)
   {
      wt[i]=bt[i-1]+wt[i-1];
   }
}

void findTurnAroundTime(int pro[], int n,int bt[], int wt[],int tat[])
{
   for(int i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];
   }
}

void findAvgTime(int pro[], int n, int bt[])
{
   int wt[n],tat[n],total_wt=0,total_tat=0;
   findWaitingTime(pro,n,bt,wt);
   findTurnAroundTime(pro,n,bt,wt,tat);

   // Calculating avg waiting time and turn around time
   cout<<"Processes    "<<"Burst time  "<<"Waiting time    "<<"Turn Around time
"<<endl;
```

```
    for(int i=0;i<n;i++)
    {
        total_wt += wt[i];
        total_tat += tat[i];
        cout<<i+1<<"            "<<bt[i]<<"             "<<wt[i]<<"             "<<tat[i]<<endl;
    }
    cout<<"Average Waiting Time "<<float(total_wt)/float(n)<<endl;
    cout<<"Average Turn Around Time "<<float(total_tat)/float(n)<<endl;

}

int main(){
    int processes[]={1,2,3};
    int n = sizeof processes/sizeof processes[0];
    int burst_time[]={10,5,8};
    findAvgTime(processes,n,burst_time);
    return 0;
}
```



```
/*******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python,
C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, C
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************
#include <iostream>

using namespace std;

void findWaitingTime(int pro[], int n,int bt[], int wt[])
{
    wt[0]=0;
    for(int i=1;i<n;i++)
```

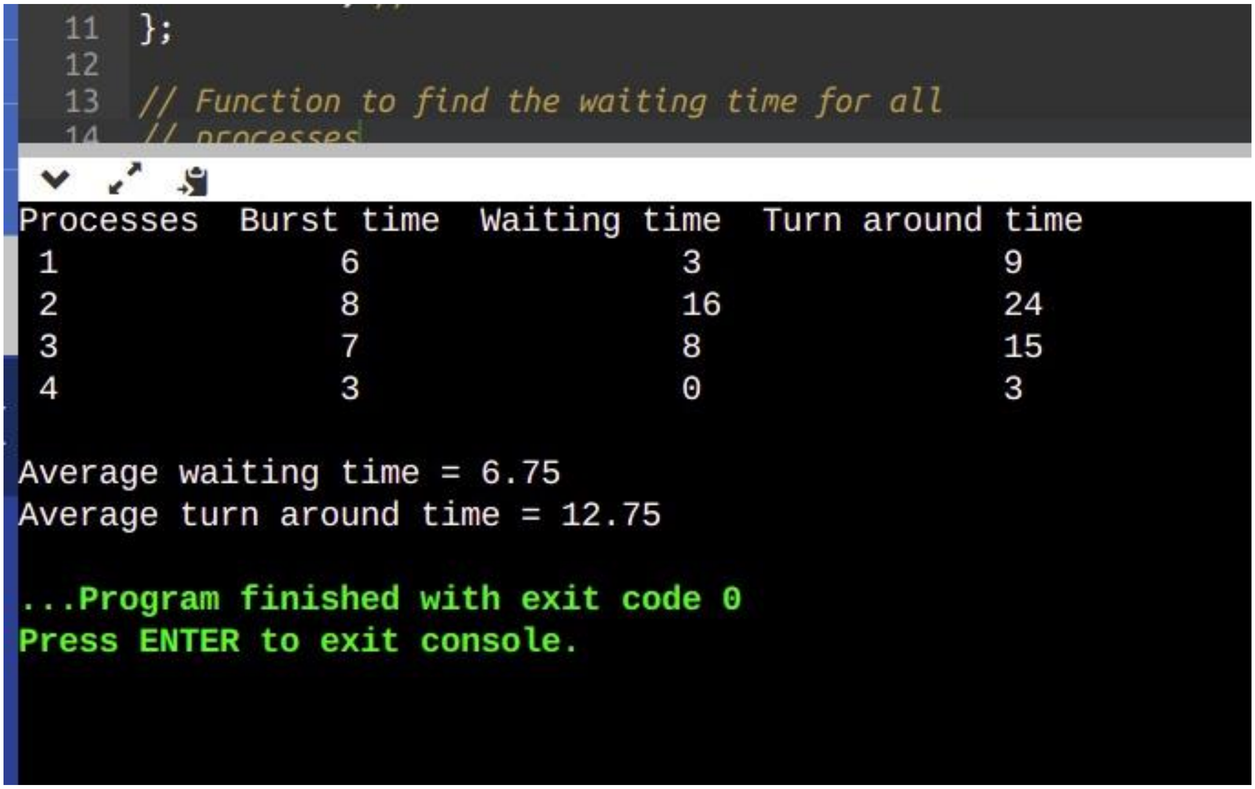
```
Processes    Burst time  Waiting time    Turn Around time
1            10             0             10
2            5             10             15
3            8             15             23
Average Waiting Time 8.33333
Average Turn Around Time 16


...Program finished with exit code 0
Press ENTER to exit console.
```

2. SJF Pre-emptive

```
// C++ program to implement Shortest Remaining Time First
// Shortest Remaining Time First (SRTF)

#include <bits/stdc++.h>
using namespace std;

struct Process {
        int pid; // Process ID
        int bt; // Burst Time
        int art; // Arrival Time
};

// Function to find the waiting time for all
// processes
void findWaitingTime(Process proc[], int n, int wt[])
{
        int rt[n];

        // Copy the burst time into rt[]
        for (int i = 0; i < n; i++)
                rt[i] = proc[i].bt;

        int complete = 0, t = 0, minm = INT_MAX;
        int shortest = 0, finish_time;
        bool check = false;

        // Process until all processes gets
        // completed
        while (complete != n) {

                // Find process with minimum
                // remaining time among the
                // processes that arrives till the
                // current time`
                for (int j = 0; j < n; j++) {
                        if ((proc[j].art <= t) &&
                        (rt[j] < minm) && rt[j] > 0) {
                                minm = rt[j];
                                shortest = j;
                                check = true;
```

```
            }
        }

        if (check == false) {
            t++;
            continue;
        }

        // Reduce remaining time by one
        rt[shortest]--;

        // Update minimum
        minm = rt[shortest];
        if (minm == 0)
            minm = INT_MAX;

        // If a process gets completely
        // executed
        if (rt[shortest] == 0) {

            // Increment complete
            complete++;
            check = false;

            // Find finish time of current
            // process
            finish_time = t + 1;

            // Calculate waiting time
            wt[shortest] = finish_time -proc[shortest].bt-proc[shortest].art;

            if (wt[shortest] < 0)
                wt[shortest] = 0;
        }
        // Increment time
        t++;
    }
}

// Function to calculate turn around time
void findTurnAroundTime(Process proc[], int n, int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n; i++)
        tat[i] = proc[i].bt + wt[i];
}

// Function to calculate average time
void findavgTime(Process proc[], int n)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    // Function to find waiting time of all
    // processes
    findWaitingTime(proc, n, wt);

    // Function to find turn around time for
    // all processes
    findTurnAroundTime(proc, n, wt, tat);

    // Display processes along with all
    // details
    cout << "Processes "
        << " Burst time "
        << " Waiting time "
        << " Turn around time\n";

    // Calculate total waiting time and
    // total turnaround time
    for (int i = 0; i < n; i++) {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << " " << proc[i].pid << "\t\t"
            << proc[i].bt << "\t\t " << wt[i]
            << "\t\t " << tat[i] << endl;
    }

    cout << "\nAverage waiting time = "
        << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "
        << (float)total_tat / (float)n;
}

// Driver code
int main()
```

```
{
	Process proc[] = { { 1, 6, 1 }, { 2, 8, 1 },{ 3, 7, 2 }, { 4, 3, 3 } };
	int n = sizeof(proc) / sizeof(proc[0]);

	findavgTime(proc, n);
	return 0;
}
```



```
};

// Function to find the waiting time for all
// processes
```

```
Processes  Burst time  Waiting time  Turn around time
 1              6               3               9
 2              8               16              24
 3              7               8               15
 4              3               0               3

Average waiting time = 6.75
Average turn around time = 12.75

...Program finished with exit code 0
Press ENTER to exit console.
```

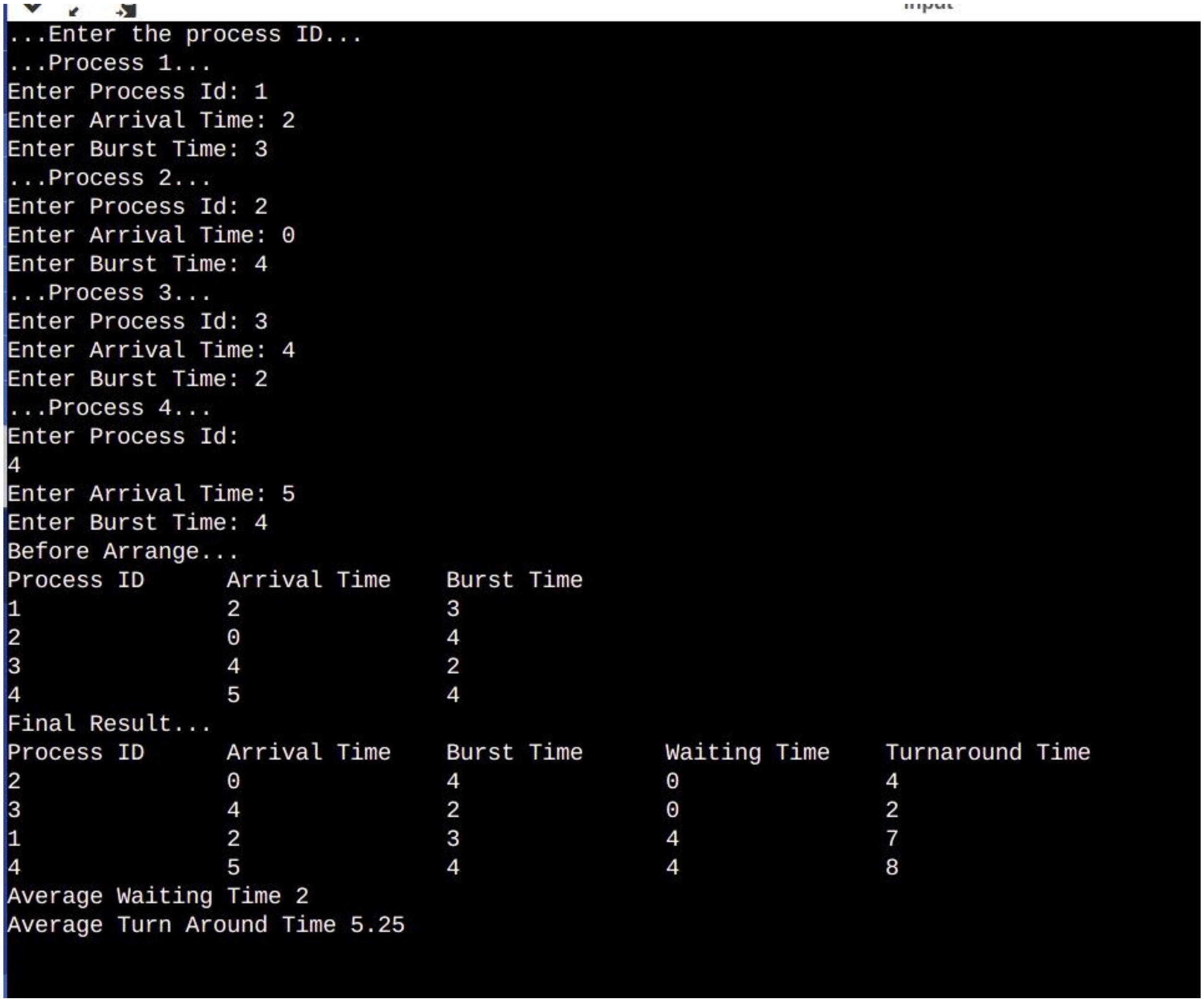
SJF Non-PreEmptive

```
// C++ program to implement Shortest Job first with Arrival
// Time
#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);
int wt1=0;
for(int i=0;i<num;i++)
{
   wt1+=mat[i][4];
}
int tat1=0;
for(int i=0;i<num;i++)
{
   tat1+=mat[i][5];
}
float avgTurnAroundTime = float(tat1)/float(num);
float avgWaitTime = float(wt1)/float(num);
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";
        }
        cout<<"Average Waiting Time "<< avgWaitTime<<endl;
        cout<<"Average Turn Around Time "<< avgTurnAroundTime<<endl;

}
```



```
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 2
Enter Burst Time: 3
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 0
Enter Burst Time: 4
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 4
Enter Burst Time: 2
...Process 4...
Enter Process Id:
4
Enter Arrival Time: 5
Enter Burst Time: 4
Before Arrange...
Process ID      Arrival Time    Burst Time
1               2               3
2               0               4
3               4               2
4               5               4
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
2               0               4               0               4
3               4               2               0               2
1               2               3               4               7
4               5               4               4               8
Average Waiting Time 2
Average Turn Around Time 5.25
```