BGC Trust University Bangladesh

Department of Computer Science & Engineering

BGC Bidyanagar, Chandanaish, Chattogram – 4381

Assignment

Assignment No:2	02
Assignment title	Implementation of FCFS,SJF,Round Robin algorithm
Course Code	07-0613-OS503
Course Title	Operating Systems
Date of Submission	01.05.25

SUBMITTED BY	SUBMITTED TO
Md. Mizan Uddin Chowdhury Raihan ID: 230240108 Session: Jan-June, 2025 Semester (Section): 5th(B)	Rashed Miah Adjunct Lecturer Department of CSE BGC Trust University Bangladesh

Signature

First come first served :

```
#include <stdio.h>

int processID[20], waitingTime[20], turnaroundTime[20], burstTime[20], totalProcesses;

void computeWaitingTime() {
    waitingTime[0] = 0;
    for (int i = 1; i < totalProcesses; i++) {
        waitingTime[i] = waitingTime[i - 1] + burstTime[i - 1];
    }
}

void computeTurnaroundTime() {
    for (int i = 0; i < totalProcesses; i++) {
        turnaroundTime[i] = burstTime[i] + waitingTime[i];
    }
}

int main() {
    printf("Enter the number of processes: ");
    scanf("%d", &totalProcesses);

    for (int i = 0; i < totalProcesses; i++) {
        processID[i] = i + 1;
        printf("Enter burst time for process %d: ", processID[i]);
        scanf("%d", &burstTime[i]);
    }

    computeWaitingTime();
    computeTurnaroundTime();

    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < totalProcesses; i++) {
        printf("%d\t\t\t%d\t\t\t%d\t\t\t%d\n", processID[i], burstTime[i], waitingTime[i],
turnaroundTime[i]);
    }

    return 0;
}
```

Output

```
Enter the number of processes: 4
Enter burst time for process 1: 3
Enter burst time for process 2: 6
Enter burst time for process 3: 8
Enter burst time for process 4: 4

Process     Burst Time  Waiting Time    Turnaround Time
1               3               0               3
2               6               3               9
3               8               9               17
4               4               17              21


=== Code Execution Successful ===
```

Shortest Job First :

```
#include <stdio.h>

int procID[20], waitTime[20], turnTime[20], burst[20], total;

void sortByBurst() {
   for (int i = 0; i < total - 1; i++) {
      for (int j = 0; j < total - i - 1; j++) {
         if (burst[j] > burst[j + 1]) {
            int temp = burst[j];
            burst[j] = burst[j + 1];
            burst[j + 1] = temp;

            temp = procID[j];
            procID[j] = procID[j + 1];
            procID[j + 1] = temp;
         }
      }
   }
}

void computeWaitingTime() {
   waitTime[0] = 0;
   for (int i = 1; i < total; i++) {
      waitTime[i] = waitTime[i - 1] + burst[i - 1];
   }
}

void computeTurnaroundTime() {
   for (int i = 0; i < total; i++) {
      turnTime[i] = burst[i] + waitTime[i];
   }
}

int main() {
   printf("Enter the number of processes: ");
   scanf("%d", &total);

   for (int i = 0; i < total; i++) {
      procID[i] = i + 1;
      printf("Enter burst time for process %d: ", procID[i]);
      scanf("%d", &burst[i]);
   }

   sortByBurst();
   computeWaitingTime();
   computeTurnaroundTime();
```

```
printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time\n");
   for (int i = 0; i < total; i++) {
      printf("%d\t\t\t\t%d\t\t\t\t%d\t\t\t\t%d\n", procID[i], burst[i], waitTime[i], turnTime[i]);
   }

   return 0;
}
```

Output

```
Enter the number of processes: 4
Enter burst time for process 1: 6
Enter burst time for process 2: 2
Enter burst time for process 3: 6
Enter burst time for process 4: 4

Process     Burst Time  Waiting Time    Turnaround Time
2               2               0               2
4               4               2               6
1               6               6               12
3               6               12              18


=== Code Execution Successful ===
```

Round Robin:

```
#include <stdio.h>

int processID[20], burstTime[20], waitingTime[20], turnaroundTime[20], totalProcesses;

void roundRobinScheduling(int timeQuantum) {
   int remainingTime[20], currentTime = 0, allDone;

   for (int i = 0; i < totalProcesses; i++) {
      remainingTime[i] = burstTime[i];
      waitingTime[i] = 0;
   }

   do {
      allDone = 1;
      for (int i = 0; i < totalProcesses; i++) {
         if (remainingTime[i] > 0) {
            allDone = 0;

            if (remainingTime[i] > timeQuantum) {
               currentTime += timeQuantum;
               remainingTime[i] -= timeQuantum;
            } else {
               currentTime += remainingTime[i];
               waitingTime[i] = currentTime - burstTime[i];
               remainingTime[i] = 0;
            }
```

```
        }
      }
    } while (!allDone);

    for (int i = 0; i < totalProcesses; i++) {
      turnaroundTime[i] = burstTime[i] + waitingTime[i];
    }
}

int main() {
    int quantum;

    printf("Enter the number of processes: ");
    scanf("%d", &totalProcesses);

    for (int i = 0; i < totalProcesses; i++) {
      processID[i] = i + 1;
      printf("Enter burst time for process %d: ", processID[i]);
      scanf("%d", &burstTime[i]);
    }

    printf("Enter time quantum: ");
    scanf("%d", &quantum);

    roundRobinScheduling(quantum);

    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < totalProcesses; i++) {
      printf("%d\t\t\t%d\t\t\t%d\t\t\t%d\n", processID[i], burstTime[i], waitingTime[i],
turnaroundTime[i]);
    }

    return 0;
}
```

Output

```
Enter the number of processes: 4
Enter burst time for process 1: 3
Enter burst time for process 2: 8
Enter burst time for process 3: 5
Enter burst time for process 4: 2
Enter time quantum: 5

Process     Burst Time  Waiting Time    Turnaround Time
1               3               0               3
2               8               10              18
3               5               8               13
4               2               13              15


=== Code Execution Successful ===
```