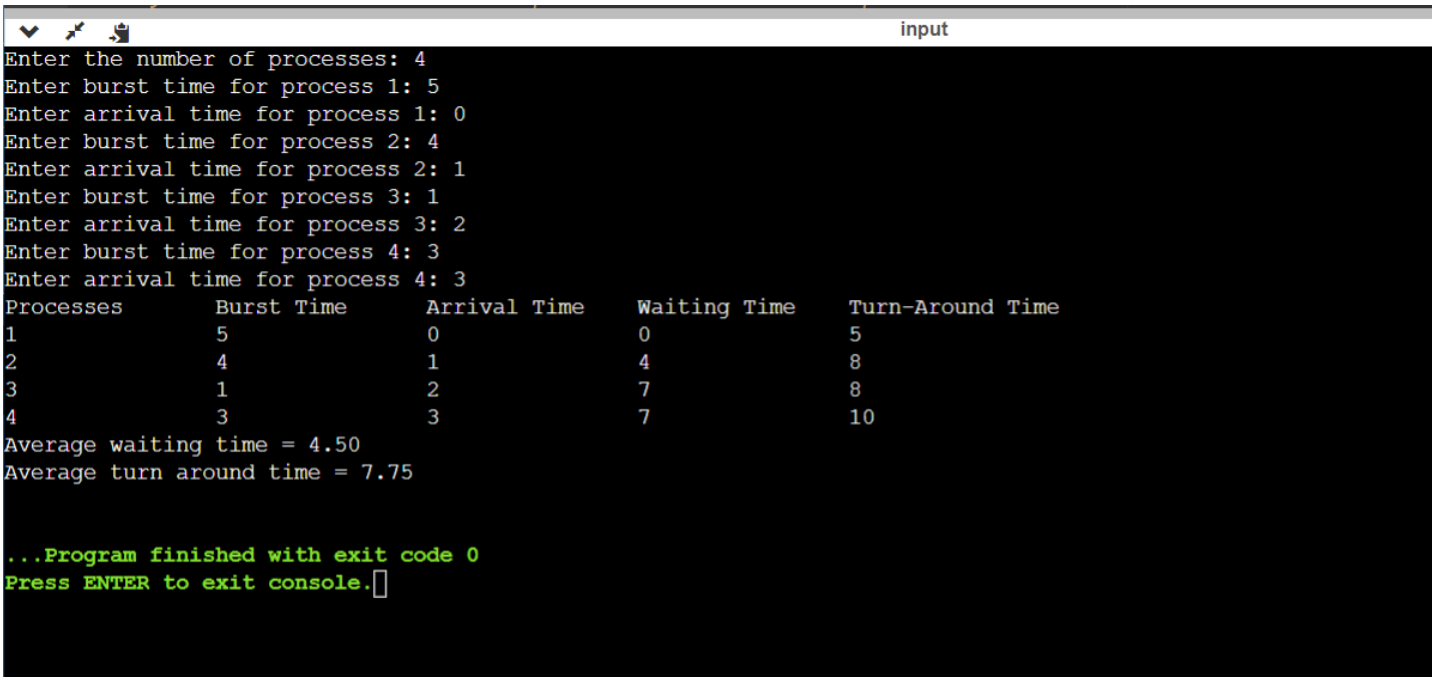
Assignment -3

a. FCFS

```
#include <stdio.h>

void calculateWaitingTime(int processes[], int n, int burstTime[], int
waitingTime[], int arrivalTime[]) {
  int serviceTime[n];
  serviceTime[0] = 0;
  waitingTime[0] = 0;

  for (int i = 1; i < n; i++) {
    serviceTime[i] = serviceTime[i - 1] + burstTime[i - 1];

    waitingTime[i] = serviceTime[i] - arrivalTime[i];
    if (waitingTime[i] < 0) {
      waitingTime[i] = 0;
    }
  }
}

void calculateTurnAroundTime(int processes[], int n, int burstTime[], int
waitingTime[], int turnAroundTime[]) {
  for (int i = 0; i < n; i++) {
    turnAroundTime[i] = burstTime[i] + waitingTime[i];
  }
}

void calculateAverageTime(int processes[], int n, int burstTime[], int
arrivalTime[]) {
  int waitingTime[n], turnAroundTime[n], totalWaitingTime = 0,
totalTurnAroundTime = 0;

  calculateWaitingTime(processes, n, burstTime, waitingTime,
arrivalTime);
  calculateTurnAroundTime(processes, n, burstTime, waitingTime,
turnAroundTime);
```

```
  printf("Processes\tBurst Time\tArrival Time\tWaiting
Time\tTurn-Around Time\n");
  for (int i = 0; i < n; i++) {
    totalWaitingTime += waitingTime[i];
    totalTurnAroundTime += turnAroundTime[i];
    printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, burstTime[i],
arrivalTime[i], waitingTime[i], turnAroundTime[i]);
  }

  printf("Average waiting time = %.2f\n", (float)totalWaitingTime /
(float)n);
  printf("Average turn around time = %.2f\n", (float)totalTurnAroundTime
/ (float)n);
}

int main() {
  int n;

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

  int processes[n], burstTime[n], arrivalTime[n];

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

  calculateAverageTime(processes, n, burstTime, arrivalTime);

  return 0;
}
```

OUTPUT



b. SJF

```
#include <stdio.h>

void calculateWaitingTime(int processes[], int n, int burstTime[], int
waitingTime[]) {
  waitingTime[0] = 0;

  for (int i = 1; i < n; i++) {
    waitingTime[i] = burstTime[i - 1] + waitingTime[i - 1];
  }
}
void calculateTurnAroundTime(int processes[], int n, int
burstTime[], int waitingTime[], int turnAroundTime[]) {

  for (int i = 0; i < n; i++) {
    turnAroundTime[i] = burstTime[i] + waitingTime[i];
  }
}
void sortProcessesByBurstTime(int processes[], int n, int
burstTime[]) {
  for (int i = 0; i < n; i++) {
```

```
      for (int j = i + 1; j < n; j++) {
        if (burstTime[i] > burstTime[j]) {
                    int temp = burstTime[i];
          burstTime[i] = burstTime[j];
          burstTime[j] = temp;
                  temp = processes[i];
          processes[i] = processes[j];
          processes[j] = temp;
        }
      }
    }
}

void calculateAverageTime(int processes[], int n, int burstTime[]) {
    int waitingTime[n], turnAroundTime[n], totalWaitingTime = 0,
totalTurnAroundTime = 0;
      calculateWaitingTime(processes, n, burstTime, waitingTime);

      calculateTurnAroundTime(processes, n, burstTime, waitingTime,
turnAroundTime);
    printf("Process ID\tBurst Time\tWaiting Time\tTurnaround
Time\n");
    for (int i = 0; i < n; i++) {
       totalWaitingTime += waitingTime[i];
       totalTurnAroundTime += turnAroundTime[i];
       printf("%d\t\t%d\t\t%d\t\t%d\n", processes[i], burstTime[i],
waitingTime[i], turnAroundTime[i]);
    }
    printf("Average waiting time = %.2f\n", (float)totalWaitingTime /
(float)n);
    printf("Average turnaround time = %.2f\n",
(float)totalTurnAroundTime / (float)n);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int processes[n], burstTime[n];
     for (int i = 0; i < n; i++) {
```

```
    printf("Enter burst time for process %d: ", i+1);
    scanf("%d", &burstTime[i]);
    processes[i] = i+1; // Process ID
  }

  sortProcessesByBurstTime(processes, n, burstTime);

  calculateAverageTime(processes, n, burstTime);

  return 0;
}
```

Output

```
Output
/tmp/2PKA8HzqLr.o
Enter the number of processes: 4
Enter burst time for process 1: 3
Enter burst time for process 2: 2
Enter burst time for process 3: 5
Enter burst time for process 4: 7
Process ID  Burst Time  Waiting Time    Turnaround Time
2       2       0       2
1       3       2       5
3       5       5       10
4       7       10      17
Average waiting time = 4.25
Average turnaround time = 8.50
```

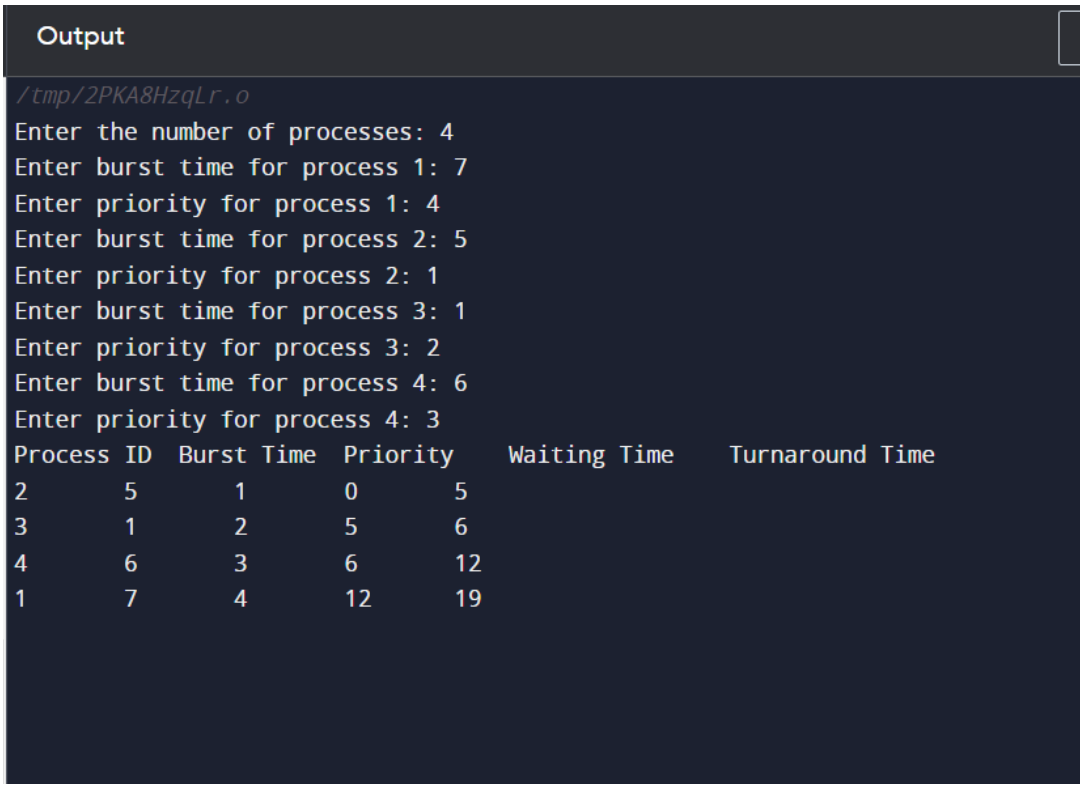
c.Priority

```
#include <stdio.h>

typedef struct process {
```

```
    int id;
    int burstTime;
    int priority;
    int waitingTime;
    int turnaroundTime;
} Process;

void sortProcessesByPriority(Process p[], int n) {
    // Simple selection sort
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (p[i].priority > p[j].priority ||
                (p[i].priority == p[j].priority && p[i].id > p[j].id)) {
                Process temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
}

void calculateWaitingTime(Process p[], int n) {
    p[0].waitingTime = 0; // first process has no waiting time
    for (int i = 1; i < n; i++) {
        p[i].waitingTime = p[i - 1].waitingTime + p[i - 1].burstTime;
    }
}

void calculateTurnaroundTime(Process p[], int n) {
    for (int i = 0; i < n; i++) {
        p[i].turnaroundTime = p[i].waitingTime + p[i].burstTime;
    }
}

void printTimes(Process p[], int n) {
    printf("Process ID\tBurst Time\tPriority\tWaiting
Time\tTurnaround Time\n");
```

```
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].id,
p[i].burstTime, p[i].priority, p[i].waitingTime,
p[i].turnaroundTime);
    }
}

int main() {
    int n;

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

    Process p[n];

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

    sortProcessesByPriority(p, n);
    calculateWaitingTime(p, n);
    calculateTurnaroundTime(p, n);
    printTimes(p, n);

    return 0;
}
```

OUTPUT


Output
/tmp/2PKA8HzqLr.o
Enter the number of processes: 4
Enter burst time for process 1: 7
Enter priority for process 1: 4
Enter burst time for process 2: 5
Enter priority for process 2: 1
Enter burst time for process 3: 1
Enter priority for process 3: 2
Enter burst time for process 4: 6
Enter priority for process 4: 3
Process ID Burst Time Priority Waiting Time Turnaround Time
2 5 1 0 5
3 1 2 5 6
4 6 3 6 12
1 7 4 12 19