FCFS

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
struct process {
   char pname[4];
   int bt, wt, tt, ct;
} p[10];
void main() {
   int n, i;
   float avgwt = 0, avgtt = 0;
 printf("Enter the number of processes: ");
   scanf("%d", &n);
 for (i = 0; i < n; i++) {
       printf("Enter the process name: ");
       scanf("%s", p[i].pname);
       printf("Enter the burst time: ");
       scanf("%d", &p[i].bt);
   }
p[0].wt = 0;
   for (i = 1; i < n; i++) {
       p[i].wt = p[i - 1].bt + p[i - 1].wt;
   }
for (i = 0; i < n; i++) {
       p[i].tt = p[i].wt + p[i].bt;
       p[i].ct = p[i].tt;
   }
printf("\nProcess Name\tBurst Time\tWaiting Time\tTurnaround
Time\tCompletionTime\n");
   for (i = 0; i < n; i++) {
       printf("%s\t\t%d\t\t%d\t\t%d\t\t%d\n",
            p[i].pname, p[i].bt, p[i].wt, p[i].tt, p[i].ct);
   }
 for (i = 0; i < n; i++) {
       avgwt += p[i].wt;
   }
   avgwt /= n;
   printf("Average Waiting Time = %f\n", avgwt);

     for (i = 0; i < n; i++) {
```

```
 avgtt += p[i].tt;
    }
    avgtt /= n;
    printf("Average Turnaround Time = %f\n", avgtt);
printf("\nGantt Chart:\n");
    int totalBurstTime = 0;
    for (i = 0; i < n; i++) {
        totalBurstTime += p[i].bt;
        printf("%s --> (%d)", p[i].pname, totalBurstTime);
        }
    printf("\n");
}
```

OUTPUT

```
Enter the number of processes: 5
Enter the process name: p0
Enter the burst time: 2
Enter the process name: p1
Enter the burst time: 6
Enter the process name: p2
Enter the burst time: 4
Enter the process name: p3
Enter the burst time: 9
Enter the process name: p4
Enter the burst time: 12

Process Name    Burst Time      Waiting Time    Turnaround Time CompletionTime
p0              2               0               2               2
p1              6               2               8               8
p2              4               8               12              12
p3              9               12              21              21
p4              12              21              33              33
Average Waiting Time = 8.600000
Average Turnaround Time = 15.200000

Gantt Chart:
p0 --> (2)p1 --> (8)p2 --> (12)p3 --> (21)p4 --> (33)
```

SJF

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct process {
   char pname[4];
   int bt,wt,tt,ct;
} p[10], temp;
void main() {
int n, i;
float avgwt = 0, avgtt = 0;
printf("Enter the number of processes: ");
scanf("%d", &n);

for (i = 0; i < n; i++) {
printf("Enter the process name: ");
scanf("%s", p[i].pname);
printf("Enter the burst time: ");
scanf("%d", &p[i].bt);
}
 for (i = 0; i < n - 1; i++) {
       for (int j = 0; j < n - i - 1; j++) {
          if (p[j].bt > p[j + 1].bt) {
             temp = p[j];
             p[j] = p[j + 1];
             p[j + 1] = temp;
          }
       }
   }
p[0].wt = 0;
   for (i = 1; i < n; i++) {
      p[i].wt = p[i - 1].bt + p[i - 1].wt;
   }

   for (i = 0; i < n; i++) {
      p[i].tt = p[i].wt + p[i].bt;
 p[i].ct = p[i].tt;
   }
```

```
   printf("\nProcess Name\tBurst Time\tWaiting Time\tTurnaround Time\tCompletion
Time\n");
   for (i = 0; i < n; i++) {
      printf("%s\t\t%d\t\t%d\t\t%d\t\t%d\n",
           p[i].pname, p[i].bt, p[i].wt, p[i].tt, p[i].ct);
   }
for (i = 0; i < n; i++) {
      avgwt += p[i].wt;
   }
   avgwt /= n;
   printf("Average Waiting Time = %f\n", avgwt);
for (i = 0; i < n; i++) {
      avgtt += p[i].tt;
   }
   avgtt /= n;
   printf("Average Turnaround Time = %f\n", avgtt);
 printf("\nGantt Chart:\n");
   int totalBurstTime = 0;
   for (i = 0; i < n; i++) {
      totalBurstTime += p[i].bt;
      printf("%s --> (%d)", p[i].pname, totalBurstTime);

   }
}
```

OUTPUT

```
Enter the number of processes: 5
Enter the process name: p1
Enter the burst time: 7
Enter the process name: p2
Enter the burst time: 3
Enter the process name: p3
Enter the burst time: 2
Enter the process name: p4
Enter the burst time: 10
Enter the process name: p5
Enter the burst time: 8

Process Name    Burst Time      Waiting Time    Turnaround Time Completion Time
p3              2               0               2               2
p2              3               2               5               5
p1              7               5               12              12
p5              8               12              20              20
p4              10              20              30              30
Average Waiting Time = 7.800000
Average Turnaround Time = 13.800000

Gantt Chart:
p3 --> (2)p2 --> (5)p1 --> (12)p5 --> (20)p4 --> (30)
```

PRIORITY

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct process {
   char pname[4];
   int at, bt, wt, tt, ct,rt,priority;
} p[10], temp;

void main() {
   int n, i, j;
   float avgwt = 0, avgtt = 0, avgrt = 0;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
  for (i = 0; i < n; i++) {
      printf("Enter the process name: ");
      scanf("%s", p[i].pname);
      printf("Enter the arrival time: ");
      scanf("%d", &p[i].at);
      printf("Enter the burst time: ");
      scanf("%d", &p[i].bt);
      printf("Enter the priority: ");
      scanf("%d", &p[i].priority);
      p[i].rt = -1;
   }
 for (i = 0; i < n - 1; i++) {
      for (j = 0; j < n - i - 1; j++) {
         if (p[j].at > p[j + 1].at ||
           (p[j].at == p[j + 1].at && p[j].priority > p[j + 1].priority)) {
            temp = p[j];
            p[j] = p[j + 1];
            p[j + 1] = temp;
         }
      }
   }
int currentTime = 0;
   int completed = 0;
   int isCompleted[10] = {0};
```

```
    while (completed < n) {
        int idx = -1;
        for (i = 0; i < n; i++) {
            if (!isCompleted[i] && p[i].at <= currentTime) {
                if (idx == -1 || p[i].priority < p[idx].priority) {
                    idx = i;
                }
            }
        }
if (idx != -1) {

            if (p[idx].rt == -1) {
                p[idx].rt = currentTime - p[idx].at;
            }

            currentTime += p[idx].bt;
            p[idx].ct = currentTime;
            p[idx].wt = currentTime - p[idx].at - p[idx].bt;
            completed++;
            isCompleted[idx] = 1;
        } else {
            currentTime++;
        }
    }
 for (i = 0; i < n; i++) {
        p[i].tt = p[i].wt + p[i].bt;
    }
  printf("\nPName\tPr\tAT\tBT\tWT\tTT\tCT\tRT\n");
    for (i = 0; i < n; i++) {
        printf("%s\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
              p[i].pname, p[i].priority, p[i].at, p[i].bt, p[i].wt, p[i].tt, p[i].ct, p[i].rt);
    }
for (i = 0; i < n; i++) {
        avgwt += p[i].wt;
    }
    avgwt /= n;
    printf("Average Waiting Time = %f\n", avgwt);

    for (i = 0; i < n; i++) {
        avgtt += p[i].tt;
```

```
    }
    avgtt /= n;
    printf("Average Turnaround Time = %f\n", avgtt);

    for (i = 0; i < n; i++) {
        avgrt += p[i].rt;
    }
    avgrt /= n;
    printf("Average Response Time = %f\n", avgrt);
  printf("\nGantt Chart:\n");
    int totalBurstTime = 0;
    for (i = 0; i < n; i++) {
        totalBurstTime += p[i].bt;
        printf("%s --> (%d)", p[i].pname, totalBurstTime);
       }
    printf("\n");
}
```

OUTPUT

```
Enter the number of processes: 7
Enter the process name: p1
Enter the arrival time: 0
Enter the burst time: 8
Enter the priority: 3
Enter the process name: p2
Enter the arrival time: 1
Enter the burst time: 2
Enter the priority: 4
Enter the process name: p3
Enter the arrival time: 3
Enter the burst time: 4
Enter the priority: 4
Enter the process name: p4
Enter the arrival time: 4
Enter the burst time: 1
Enter the priority: 5
Enter the process name: p5
Enter the arrival time: 6
Enter the burst time: 6
Enter the priority: 2
Enter the process name: p6
Enter the arrival time: 6
Enter the burst time: 5
Enter the priority: 6
Enter the process name: p7
Enter the arrival time: 10
Enter the burst time: 1
Enter the priority: 1
```

```
PName   Pr      AT      BT      WT      TT      CT      RT
p1      3       0       8       0       8       8       0
p2      4       1       2       14      16      17      14
p3      4       3       4       14      18      21      14
p4      5       4       1       17      18      22      17
p5      2       6       6       2       8       14      2
p6      6       6       5       16      21      27      16
p7      1       10      1       4       5       15      4
Average Waiting Time = 9.571428
Average Turnaround Time = 13.428572
Average Response Time = 9.571428

Gantt Chart:
p1 --> (8)p2 --> (10)p3 --> (14)p4 --> (15)p5 --> (21)p6 --> (26)p7 --> (27)
```

ROUND ROBIN

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct process {
   char pname[4];
   int at, bt, wt, tt, ct, rt, remaining_bt;
} p[10], temp;

void main() {
   int n, i;
   float avgwt = 0, avgtt = 0, avgrt = 0;
   int timeQuantum;

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

   for (i = 0; i < n; i++) {
      printf("Enter the process name: ");
      scanf("%s", p[i].pname);
      printf("Enter the arrival time: ");
      scanf("%d", &p[i].at);
      printf("Enter the burst time: ");
      scanf("%d", &p[i].bt);
      p[i].remaining_bt = p[i].bt;
      p[i].rt = -1;
   }

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

   int currentTime = 0;
   int completed = 0;
   int isCompleted[10] = {0};

   while (completed < n) {
      int foundProcess = 0;
```

```
      for (i = 0; i < n; i++) {
         if (!isCompleted[i] && p[i].at <= currentTime) {
            foundProcess = 1;
          if (p[i].remaining_bt > 0) {
               if (p[i].rt == -1) {
                  p[i].rt = currentTime - p[i].at;
               }
 if (p[i].remaining_bt > timeQuantum) {
                  currentTime += timeQuantum;
                  p[i].remaining_bt -= timeQuantum;
               } else {
                  currentTime += p[i].remaining_bt;
                  p[i].ct = currentTime;
                  p[i].wt = currentTime - p[i].at - p[i].bt;
                  completed++;
                  isCompleted[i] = 1;
                  p[i].remaining_bt = 0;
            }
         }
      }
 if (!foundProcess) {
         currentTime++;
      }
   }
for (i = 0; i < n; i++) {
      p[i].tt = p[i].wt + p[i].bt;
   }
printf("\nPName\tAT\tBT\tWT\tTT\tCT\tRT\n");
   for (i = 0; i < n; i++) {
      printf("%s\t%d\t%d\t%d\t%d\t%d\t%d\n",
            p[i].pname, p[i].at, p[i].bt, p[i].wt, p[i].tt, p[i].ct, p[i].rt);
   }
for (i = 0; i < n; i++) {
      avgwt += p[i].wt;
   }
   avgwt /= n;
   printf("Average Waiting Time = %f\n", avgwt);

   for (i = 0; i < n; i++) {
      avgtt += p[i].tt;
   }
```

```
    avgtt /= n;
    printf("Average Turnaround Time = %f\n", avgtt);

    for (i = 0; i < n; i++) {
        avgrt += p[i].rt;
    }
    avgrt /= n;
    printf("Average Response Time = %f\n", avgrt);
printf("\nGantt Chart:\n");
    int totalBurstTime = 0;
    for (i = 0; i < n; i++) {
        totalBurstTime += p[i].bt;
        printf("%s --> (%d)", p[i].pname, totalBurstTime);
    }
  printf("\n");
}
```

OUTPUT

```
Enter the number of processes: 5
Enter the process name: p1
Enter the arrival time: 0
Enter the burst time: 8
Enter the process name: p2
Enter the arrival time: 5
Enter the burst time: 2
Enter the process name: p3
Enter the arrival time: 1
Enter the burst time: 7
Enter the process name: p4
Enter the arrival time: 6
Enter the burst time: 3
Enter the process name: p5
Enter the arrival time: 8
Enter the burst time: 5
Enter the time quantum: 3

PName    AT       BT       WT       TT       CT       RT
p1       0        8        16       24       24       0
p2       5        2        10       12       17       10
p3       1        7        17       24       25       2
p4       6        3        0        3        9        0
p5       8        5        9        14       22       1
Average Waiting Time = 10.400000
Average Turnaround Time = 15.400000
Average Response Time = 2.600000

Gantt Chart:
p1 --> (8)p2 --> (10)p3 --> (17)p4 --> (20)p5 --> (25)
```