EX NO:2A

DATE: FIRST COME FIRST SERVE SCHEDULING ALGORITHM

AIM

To implement the first come first serve scheduling algorithm.

PROGRAM

```
#include <stdio.h>


void findWaitingTime(int processes[], 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 processes[], int n, int bt[], int wt[], int tat[]) {
   for (int i = 0; i < n; i++)
      tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[]) {
   int wt[n], tat[n], total_wt = 0, total_tat = 0;
   findWaitingTime(processes, n, bt, wt);
   findTurnAroundTime(processes, n, bt, wt, tat);


   printf("Processes Burst time Waiting time Turn around time\n");
   for (int i = 0; i < n; i++) {
      total_wt += wt[i];
      total_tat += tat[i];
      printf("%d\t%d\t%d\t%d\n", (i + 1), bt[i], wt[i], tat[i]);
   }
   int avg_wt = (float)total_wt / n;
   int avg_tat = (float)total_tat / n;
   printf("Average waiting time = %d\n", avg_wt);
   printf("Average turn around time = %d\n", avg_tat);
```

Output:

```
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
................................................................................
Process executed in 2.11 seconds
Press any key to continue.
```

```
}
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;
}
```

RESULT

Thus the program to implement the first come first serve scheduling algorithm has been executed successfully.

EX NO:2B

DATE: SHORTEST JOB FIRST SCHEDULING ALGORITHM

AIM

To implement the shortest job first scheduling algorithm.

PROGRAM

```
#include<stdio.h>
#include<string.h>
int main()
{
   int et[20],at[10],n,i,j,temp,st[10],ft[10],wt[10],ta[10];
   int totwt=0,totta=0;
   float awt,ata;
   char pn[10][10],t[10];
   printf("Enter the number of process:");
   scanf("%d",&n);
   for(i=0; i<n; i++)
   {
      printf("Enter process name, arrival time& execution time:");
      scanf("%s%d%d",pn[i],&at[i],&et[i]);
   }
   for(i=0; i<n; i++)
      for(j=0; j<n; j++)
      {
         if(et[i]<et[j])
         {
            temp=at[i];
            at[i]=at[j];
            at[j]=temp;
            temp=et[i];
```

OUTPUT:

```
Enter the number of process:3
Enter process name, arrival time& execution time:2 5 7
Enter process name, arrival time& execution time:3 6 14
Enter process name, arrival time& execution time:4 7 12

Pname   arrivaltime     executiontime   waitingtime     tatime
2           5               7               0               7
4           7              12               5              17
3           6              14              18              32
Average waiting time is:7.666667
Average turnaroundtime is:18.666666
```

```
            et[i]=et[j];
            et[j]=temp;
            strcpy(t,pn[i]);
            strcpy(pn[i],pn[j]);
            strcpy(pn[j],t);
          }
       }
    for(i=0; i<n; i++)
    {
       if(i==0)
          st[i]=at[i];
       else
          st[i]=ft[i-1];
       wt[i]=st[i]-at[i];
       ft[i]=st[i]+et[i];
       ta[i]=ft[i]-at[i];
       totwt+=wt[i];
       totta+=ta[i];
    }
    awt=(float)totwt/n;
    ata=(float)totta/n;
    printf("\nPname\tarrivaltime\texecutiontime\twaitingtime\ttatime");
    for(i=0; i<n; i++)
       printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],wt[i],ta[i]);
    printf("\nAverage waiting time is:%f",awt);
    printf("\nAverage turnaroundtime is:%f",ata);
    return 0;
}
```

RESULT

Thus the program to implement shortest job first scheduling algorithm has been executed successfully

EX NO:2C

DATE: ROUND ROBIN SCHEDULING ALGORITHM

AIM

To implement the round robin scheduling algorithm.

PROGRAM

```
#include <stdio.h>
#include <conio.h>

void main() {
   int i, NOP, sum = 0, count = 0, y, quant, wt = 0, tat = 0, at[10], bt[10], temp[10];
   float avg_wt, avg_tat;
   printf("Total number of processes in the system: ");
   scanf("%d", &NOP);
   y = NOP;

   for (i = 0; i < NOP; i++) {
      printf("\nEnter the Arrival and Burst time of Process[%d]\n", i + 1);
      printf("Arrival time: ");
      scanf("%d", &at[i]);
      printf("\nBurst time: ");
      scanf("%d", &bt[i]);
      temp[i] = bt[i];
   }

   printf("Enter the Time Quantum for the process: ");
   scanf("%d", &quant);

   printf("\nProcess No\tBurst Time\tTAT\tWaiting Time");
   for (sum = 0, i = 0; y != 0;) {
      if (temp[i] <= quant && temp[i] > 0) {
         sum += temp[i];
```

OUTPUT

```
 Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  8

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       1

Burst time is:  5

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       2

Burst time is:  10

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       3

Burst time is:  11
Enter the Time Quantum for the process:         6

 Process No              Burst Time              TAT             Waiting Time

 Process No[2]           5               10              5
 Process No[1]           8               25              17
 Process No[3]           10              27              17
 Process No[4]           11              31              20
 Average Turn Around Time:      14.750000
 Average Waiting Time:  23.250000
```

```
            temp[i] = 0;
            count = 1;
        } else if (temp[i] > 0) {
            temp[i] -= quant;
            sum += quant;
        }
        if (temp[i] == 0 && count == 1) {
            y--;
            printf("\nProcess No[%d]\t%d\t\t%d\t\t%d", i + 1, bt[i], sum - at[i], sum - at[i] - bt[i]);
            wt += sum - at[i] - bt[i];
            tat += sum - at[i];
            count = 0;
        }
        if (i == NOP - 1) {
            i = 0;
        } else if (at[i + 1] <= sum) {
            i++;
        } else {
            i = 0;
        }
    }
    avg_wt = wt * 1.0 / NOP;
    avg_tat = tat * 1.0 / NOP;
    printf("\nAverage Turn Around Time: %f", avg_wt);
    printf("\nAverage Waiting Time: %f", avg_tat);
    getch();
}
```

RESULT

Thus the program to implement the round robin scheduling algorithm has been executed successfully.

EX NO:2D

DATE: PRIORITY SCHEDULING ALGORITHM

AIM

To implement the priority scheduling algorithm.

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
struct process {
   int process_id;
   int burst_time;
   int priority;
   int waiting_time;
   int turnaround_time;
};
void find_waiting_time(struct process proc[], int n, int wt[]) {
   wt[0] = 0;
   for (int i = 1; i < n; i++) {
      wt[i] = proc[i - 1].burst_time + wt[i - 1];
   }
}
void find_turnaround_time(struct process proc[], int n, int wt[], int tat[]) {
   for (int i = 0; i < n; i++) {
      tat[i] = proc[i].burst_time + wt[i];
   }
}
void find_average_time(struct process proc[], int n) {
   int wt[10], tat[10], total_wt = 0, total_tat = 0;
   find_waiting_time(proc, n, wt);
   find_turnaround_time(proc, n, wt, tat);
   printf("\nProcess ID\tBurst Time\tPriority\tWaiting Time\tTurnaround Time");
```

Output:

```
Enter the number of processes: 3
Enter the process ID: 1 Enter the burst time: 5 Enter the priority: 2
Enter the process ID: 2 Enter the burst time: 2 Enter the priority: 1
Enter the process ID: 3 Enter the burst time: 4 Enter the priority: 3
Process ID Burst Time Priority Waiting Time Turnaround Time 2 2 1 0 2 1 5 2 2 7 3 4 3 7 11
Average Waiting Time = 3.000000 Average Turnaround Time = 6.666667
```

```
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", proc[i].process_id, proc[i].burst_time, proc[i].priority,
wt[i], tat[i]);
    }
    printf("\n\nAverage Waiting Time = %f", (float)total_wt / n);
    printf("\nAverage Turnaround Time = %f\n", (float)total_tat / n);
}
void priority_scheduling(struct process proc[], int n) {
    for (int i = 0; i < n; i++) {
        int pos = i;
        for (int j = i + 1; j < n; j++) {
            if (proc[j].priority < proc[pos].priority) {
                pos = j;
            }
        }
        struct process temp = proc[i];
        proc[i] = proc[pos];
        proc[pos] = temp;
    }
    find_average_time(proc, n);
}
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct process proc[10];
    for (int i = 0; i < n; i++) {
        printf("\nEnter the process ID: ");
```

```
        scanf("%d", &proc[i].process_id);
        printf("Enter the burst time: ");
        scanf("%d", &proc[i].burst_time);
        printf("Enter the priority: ");
        scanf("%d", &proc[i].priority);
    }
    priority_scheduling(proc, n);
    return 0;
}
```

RESULT

Thus the program to implement priority scheduling algorithm has been executed successfully.