ATMA RAM SANATAN DHARMA COLLEGE
University of Delhi

Operating Systems

Practical File

Submitted By

Aditi sharma

ROLL NO. 21/18055

BSc(Hons) Computer Science

Submitted to

Dr. Parul Jain

Department of Computer Science

Q6. Write a program to implement FCFS scheduling algorithm.

Code:-

```
#include <stdio.h>
#include <stdlib.h>
struct process
{
 int pid;
 int burstTime;
 int arrivalTime;
 int waitingTime;
 int turnAroundTime;
};
void computeWaitingTime(struct process *processes, int processCount)
{
 processes[0].waitingTime = 0;
 for (int i = 0; i < processCount - 1; i++)
 processes[i + 1].waitingTime =
 processes[i].burstTime +
 processes[i].waitingTime;
}
void computeTurnAroundTime(struct process *processes, int processCount)
{
 for (int i = 0; i < processCount; i++)
 processes[i].turnAroundTime =
 processes[i].burstTime +
 processes[i].waitingTime -
 processes[i].arrivalTime;
}
void printAverageTimes(struct process *processes, int processCount)
{
 float totalWaitingTime = 0.0f;
 float totalTurnAroundTime = 0.0f;
 computeWaitingTime(processes, processCount);
 computeTurnAroundTime(processes, processCount);
printf("Process ID\tBurst Time\tArrival Time\tWaiting Time\tTurn Around
Time\n");
 printf("-----------------------------------------------------------");
 printf("-------------------------\n");
 for (int i = 0; i < processCount; i++)
 {
 totalWaitingTime += processes[i].waitingTime;
 totalTurnAroundTime += processes[i].turnAroundTime;
 printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
 processes[i].pid,
 processes[i].burstTime,
 processes[i].arrivalTime,
 processes[i].waitingTime,
```

```
 processes[i].turnAroundTime);
 }
 printf("\nAverage Waiting Time = %.2f",
 totalWaitingTime / processCount);
 printf("\nAverage Turn-Around time = %.2f\n",
 totalTurnAroundTime / processCount);
}
int main(void)
{
 int processCount;
 printf("Enter Number of Processes: ");
 scanf("%i", &processCount);
 struct process processes[processCount];
 for (int i = 0; i < processCount; i++)
 {
 processes[i].pid = i + 1;
 printf("Burst Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].burstTime);
 printf("Arrival Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].arrivalTime);
 }
 printf("\n");
 printAverageTimes(processes, processCount);
 return 0;
}
```

Output :-

```
~/OSpracts$ ./a.out
Enter Number of Processes: 3
Burst Time for Process 1: 10
Arrival Time for Process 1: 0
Burst Time for Process 2: 5
Arrival Time for Process 2: 0
Burst Time for Process 3: 8
Arrival Time for Process 3: 0

Process ID  Burst Time  Arrival Time    Waiting Time    TurnAround Time
--------------------------------------------------------------------------------
1       10      0       0       10
2       5       0       10      15
3       8       0       15      23

Average Waiting Time = 8.33
Average Turn-Around time = 16.00
```

Q7. Write a program to implement Round Robin scheduling algorithm.

Code:-

```
#include <stdio.h>
#include <stdlib.h>
struct process
{
 int pid;
 int burstTime;
 int arrivalTime;
 int waitingTime;
 int turnAroundTime;
};
void computeWaitingTime(struct process *processes, int processCount, int
quantum)
{
 int remainingTime[processCount];
 for (int i = 0; i < processCount; i++)
 remainingTime[i] = processes[i].burstTime;
 int time = 0;
 while (1)
 {
 int done = 1;
 for (int i = 0; i < processCount; i++)
 {
 if (remainingTime[i] > 0)
 {
 done = 0;
 if (remainingTime[i] > quantum)
 {
 time += quantum;
 remainingTime[i] -= quantum;
 }
 else
 {
 time += remainingTime[i];
 processes[i].waitingTime = time - processes[i].burstTime;
remainingTime[i] = 0;
 }
 }
 }
 if (done == 1)
 break;
 }
}
void computeTurnAroundTime(struct process *processes, int processCount)
{
 for (int i = 0; i < processCount; i++)
```

```
 processes[i].turnAroundTime =
 processes[i].burstTime +
 processes[i].waitingTime -
 processes[i].arrivalTime;
}
void printAverageTimes(struct process *processes, int processCount, int
quantum)
{
 float totalWaitingTime = 0.0f;
 float totalTurnAroundTime = 0.0f;
 computeWaitingTime(processes, processCount, quantum);
 computeTurnAroundTime(processes, processCount);
 printf("Process ID\tBurst Time\tArrival Time\tWaiting Time\tTurn Around
Time\n");
 printf("-------------------------------------------------------");
 printf("-------------------------\n");
 for (int i = 0; i < processCount; i++)
 {
 totalWaitingTime += processes[i].waitingTime;
 totalTurnAroundTime += processes[i].turnAroundTime;
 printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
 processes[i].pid,
 processes[i].burstTime,
 processes[i].arrivalTime,
 processes[i].waitingTime,
 processes[i].turnAroundTime);
 }
 printf("\nAverage Waiting Time = %.2f",
 totalWaitingTime / processCount);
 printf("\nAverage Turn-Around time = %.2f\n",
 totalTurnAroundTime / processCount);
}
int main(void)
{int processCount, quantum;
 printf("Enter Time Quantum: ");
 scanf("%i", &quantum);
 printf("Enter Number of Processes: ");
 scanf("%i", &processCount);
 struct process processes[processCount];
 for (int i = 0; i < processCount; i++)
 {
 processes[i].pid = i + 1;
 printf("Burst Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].burstTime);
 printf("Arrival Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].arrivalTime);
 }
 printf("\n");
```

```
 printAverageTimes(processes, processCount, quantum);
 return 0;
}
```

Output:-

```
~/OSpracts$ ./a.out
Enter Time Quantum: 2
Enter Number of Processes: 5
Burst Time for Process 1: 2
Arrival Time for Process 1: 0
Burst Time for Process 2: 1
Arrival Time for Process 2: 0
Burst Time for Process 3: 8
Arrival Time for Process 3: 0
Burst Time for Process 4: 4
Arrival Time for Process 4: 0
Burst Time for Process 5: 5
Arrival Time for Process 5: 0

Process ID  Burst Time  Arrival Time    Waiting Time    TurnAround Time
--------------------------------------------------------------------------------
1       2       0       0       2
2       1       0       2       3
3       8       0       12      20
4       4       0       9       13
5       5       0       13      18

Average Waiting Time = 7.20
Average Turn-Around time = 11.20
~/OSpracts$
```

Q8. Write a program to implement SJF scheduling algorithm.

Code:-

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
struct process
{
 int pid;
 int burstTime;
 int arrivalTime;
 int waitingTime;
 int turnAroundTime;
};
void computeWaitingTime(struct process *processes, int processCount)
{
 int remainingTime[processCount];
 for (int i = 0; i < processCount; i++)
 remainingTime[i] = processes[i].burstTime;
 int check = 0;
 int min = INT_MAX;
 int completionTime, time = 0;
 int complete = 0, shortest = 0;
 while (complete != processCount)
 {
 for (int j = 0; j < processCount; j++)
 {
 if ((processes[j].arrivalTime <= time) &&
 (remainingTime[j] < min) && remainingTime[j] > 0)
 {
 min = remainingTime[j];
 shortest = j;
 check = 1;
 }
   }
 if (check == 0)
 {
 time++;
 continue;
 }
 remainingTime[shortest]--;
 min = remainingTime[shortest];
 if (min == 0)
 min = INT_MAX;
 if (remainingTime[shortest] == 0)
 {
 complete++;
 check = 0;
```

```
 completionTime = time + 1;
 processes[shortest].waitingTime =
 completionTime -
 processes[shortest].burstTime -
 processes[shortest].arrivalTime;
 if (processes[shortest].burstTime < 0)
 processes[shortest].burstTime = 0;
 }
 time++;
 }
}
void computeTurnAroundTime(struct process *processes, int processCount)
{
 for (int i = 0; i < processCount; i++)
 processes[i].turnAroundTime =
 processes[i].burstTime +
 processes[i].waitingTime;
}
void printAverageTimes(struct process *processes, int processCount)
{
 float totalWaitingTime = 0.0f;
 float totalTurnAroundTime = 0.0f;
 computeWaitingTime(processes, processCount);
 computeTurnAroundTime(processes, processCount);
printf("Process ID\tBurst Time\tArrival Time\tWaiting Time\tTurn Around
Time\n");
 printf("-------------------------------------------------------");
 printf("-------------------------\n");
 for (int i = 0; i < processCount; i++)
 {
 totalWaitingTime += processes[i].waitingTime;
 totalTurnAroundTime += processes[i].turnAroundTime;
 printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
 processes[i].pid,
 processes[i].burstTime,
 processes[i].arrivalTime,
 processes[i].waitingTime,
 processes[i].turnAroundTime);
 }
 printf("\nAverage Waiting Time = %.2f",
 totalWaitingTime / processCount);
 printf("\nAverage Turn-Around time = %.2f\n",
 totalTurnAroundTime / processCount);
}
int main(void)
{
 int processCount;
 printf("Enter Number of Processes: ");
```

```
 scanf("%i", &processCount);
 struct process processes[processCount];
 for (int i = 0; i < processCount; i++)
 {
 processes[i].pid = i + 1;
 printf("Burst Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].burstTime);
 printf("Arrival Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].arrivalTime);
 }
 printf("\n");
 printAverageTimes(processes, processCount);
 return 0;
}
```

Output :-

```
~/OSpracts$ ./a.out
Enter Number of Processes: 4
Burst Time for Process 1: 6
Arrival Time for Process 1: 1
Burst Time for Process 2: 8
Arrival Time for Process 2: 1
Burst Time for Process 3: 7
Arrival Time for Process 3: 2
Burst Time for Process 4: 3
Arrival Time for Process 4: 3

Process ID  Burst Time  Arrival Time    Waiting Time    TurnAround Time
--------------------------------------------------------------------------------
1       6       1       3       9
2       8       1       16      24
3       7       2       8       15
4       3       3       0       3

Average Waiting Time = 6.75
Average Turn-Around time = 12.75
~/OSpracts$
```