Experiment Number: 7

Problem Statement: CPU Scheduling Algorithms

Name: Arnav Shah Roll No. : 21

Class : AI_C Batch : B2

1) First Come First Search :-

```
#include<iostream>
using namespace std;

// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[], int at[]) {
   int service_time[n];
   service_time[0] = at[0]; // Service time for first process is its arrival time
   wt[0] = 0; // Waiting time for first process is 0

   // calculating waiting time
   for (int i = 1; i < n; i++) {
      // Calculating service time for each process
      service_time[i] = service_time[i - 1] + bt[i - 1];

      // If the current process hasn't arrived yet, wait until it arrives
      if (service_time[i] < at[i])
         service_time[i] = at[i];

      // Calculate waiting time
      wt[i] = service_time[i] - at[i];

      // If waiting time is negative, make it 0
      if (wt[i] < 0)
         wt[i] = 0;
   }
```

```
}

// Function to calculate turn around time
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
   // calculating turnaround time by adding bt[i] + wt[i]
   for (int i = 0; i < n; i++)
      tat[i] = bt[i] + wt[i];
}

// Function to calculate average time
void findavgTime(int processes[], int n, int bt[], int at[]) {
   int wt[n], tat[n], total_wt = 0, total_tat = 0;

   // Function to find waiting time of all processes
   findWaitingTime(processes, n, bt, wt, at);

   // Function to find turn around time for all processes
   findTurnAroundTime(processes, n, bt, wt, tat);

   // Display processes along with all details
   cout << "Processes " << " Arrival time " << " Burst time "
        << " Waiting time " << " Turn around time\n";

   // Calculate total waiting time and total turn around time
   for (int i = 0; i < n; i++) {
      total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
      cout << " " << processes[i] << "\t\t" << at[i] << "\t\t"
           << bt[i] << "\t " << wt[i] << "\t\t " << tat[i] << endl;
   }
```

```
    cout << "Average waiting time = "
        << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "
        << (float)total_tat / (float)n;
}

int main() {
    int n;
    cout << "Enter the number of processes: ";
    cin >> n;

    int processes[n];
    int arrival_time[n];
    int burst_time[n];

    cout << "Enter arrival time for each process:\n";
    for (int i = 0; i < n; i++) {
        cout << "Arrival time of process " << i + 1 << ": ";
        cin >> arrival_time[i];
        processes[i] = i + 1; // Assigning process IDs
    }

    cout << "Enter burst time for each process:\n";
    for (int i = 0; i < n; i++) {
        cout << "Burst time for process " << i + 1 << ": ";
        cin >> burst_time[i];
    }

    findavgTime(processes, n, burst_time, arrival_time);
    return 0;
}
```

Output :-

```
Enter the number of processes: 4
Enter arrival time for each process:
Arrival time of process 1: 2
Arrival time of process 2: 3
Arrival time of process 3: 5
Arrival time of process 4: 7
Enter burst time for each process:
Burst time for process 1: 4
Burst time for process 2: 14
Burst time for process 3: 2
Burst time for process 4: 5
Processes  Arrival time  Burst time  Waiting time  Turn around time
 1              2              4        0              4
 2              3              14       3              17
 3              5              2        15             17
 4              7              5        15             20
Average waiting time = 8.25
Average turn around time = 14.5

...Program finished with exit code 0
Press ENTER to exit console.
```

2) Shortest Job First :-

```
struct Process {

int arrival_time;

int burst_time;

int waiting_time;

};


int compare(const void *a, const void *b) {

struct Process *p1 = (struct Process *)a;

struct Process *p2 = (struct Process *)b;

return p1->burst_time - p2->burst_time;

}


int main() {

int n, i, j;

float avg_waiting_time = 0, avg_turnaround_time = 0;

printf("Enter the number of processes: ");
```

```
scanf("%d", &n);
struct Process processes[n];
for (i = 0; i< n; i++) {
printf("Enter arrival time and burst time of process %d: ", i+1);
scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
}
qsort(processes, n, sizeof(struct Process), compare);
processes[0].waiting_time = 0;
for (i = 1; i< n; i++) {
processes[i].waiting_time = 0;
for (j = 0; j <i; j++)
{

processes[i].waiting_time += processes[j].burst_time;


}


avg_waiting_time += processes[i].waiting_time;


}


avg_waiting_time /= n;


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


avg_turnaround_time += processes[i].burst_time + processes[i].waiting_time;


}


avg_turnaround_time /= n;
```

```
printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i+1, processes[i].arrival_time, processes[i].burst_time,
processes[i].waiting_time, processes[i].burst_time+processes[i].waiting_time);

}

printf("\nAverage Waiting Time: %f\n", avg_waiting_time);

printf("Average Turnaround Time: %f\n", avg_turnaround_time);

return 0;

}
```

Output :-

```
Enter the number of processes: 4
Enter arrival time and burst time of process 1: 1 3
Enter arrival time and burst time of process 2: 2 4
Enter arrival time and burst time of process 3: 1 2
Enter arrival time and burst time of process 4: 4 4

Process Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               1               2               0               2
2               1               3               2               5
3               2               4               5               9
4               4               4               9               13

Average Waiting Time: 4.000000
Average Turnaround Time: 7.250000


...Program finished with exit code 0
Press ENTER to exit console.
```

3) Round Robin :-

```
#include <iostream>
#include <climits>
using namespace std;

struct Process {
    int AT, BT, ST[20], WT, FT, TAT, pos;
};

int quant;

int main() {
    int n, i, j;

    // Taking Input
    cout << "Enter the no. of processes: ";
    cin >> n;
    Process p[n];

    cout << "Enter the quantum: " << endl;
    cin >> quant;

    cout << "Enter the process numbers: " << endl;
    for (i = 0; i < n; i++)
        cin >> p[i].pos;

    cout << "Enter the Arrival time of processes: " << endl;
    for (i = 0; i < n; i++)
        cin >> p[i].AT;

    cout << "Enter the Burst time of processes: " << endl;
```

```
for (i = 0; i < n; i++)
        cin >> p[i].BT;

// Declaring variables
int c = n, s[n][20];
float time = 0, mini = INT_MAX, b[n], a[n];

// Initializing burst and arrival time arrays
int index = -1;
for (i = 0; i < n; i++) {
        b[i] = p[i].BT;
        a[i] = p[i].AT;
        for (j = 0; j < 20; j++) {
                s[i][j] = -1;
        }
}

int tot_wt, tot_tat;
tot_wt = 0;
tot_tat = 0;
bool flag = false;

while (c != 0) {
        mini = INT_MAX;
        flag = false;

        for (i = 0; i < n; i++) {
                float p = time + 0.1;
                if (a[i] <= p && mini > a[i] && b[i] > 0) {
                        index = i;
                        mini = a[i];
```

```
                    flag = true;
            }
}

// if at =1 then loop gets out hence set flag to false
if (!flag) {
            time++;
            continue;
}

// calculating start time
j = 0;

while (s[index][j] != -1) {
            j++;
}

if (s[index][j] == -1) {
            s[index][j] = time;
            p[index].ST[j] = time;
}

if (b[index] <= quant) {
            time += b[index];
            b[index] = 0;
} else {
            time += quant;
            b[index] -= quant;
}

if (b[index] > 0) {
```

```
                a[index] = time + 0.1;
        }

        // calculating arrival, burst, final times
        if (b[index] == 0) {
                c--;
                p[index].FT = time;
                p[index].WT = p[index].FT - p[index].AT - p[index].BT;
                tot_wt += p[index].WT;
                p[index].TAT = p[index].BT + p[index].WT;
                tot_tat += p[index].TAT;
        }
} // end of while loop

// Printing output
cout << "Process number ";
cout << "Arrival time ";
cout << "Burst time ";
cout << "\tStart time";
j = 0;
while (j != 10) {
        j += 1;
        cout << " ";
}
cout << "\t\tFinal time";
cout << "\tWait Time ";
cout << "\tTurnAround Time" << endl;

for (i = 0; i < n; i++) {
        cout << p[i].pos << "\t\t";
        cout << p[i].AT << "\t\t";
```

```
            cout << p[i].BT << "\t";
            j = 0;
            int v = 0;
            while (s[i][j] != -1) {
                cout << p[i].ST[j] << " ";
                j++;
                v += 3;
            }
            while (v != 40) {
                cout << " ";
                v += 1;
            }
            cout << p[i].FT << "\t\t";
            cout << p[i].WT << "\t\t";
            cout << p[i].TAT << endl;
    }

    // Calculating average wait time and turnaround time
    double avg_wt, avg_tat;
    avg_wt = tot_wt / static_cast<double>(n);
    avg_tat = tot_tat / static_cast<double>(n);

    // Printing average wait time and turnaround time
    cout << "The average wait time is: " << avg_wt << endl;
    cout << "The average TurnAround time is: " << avg_tat << endl;

    return 0;
}
```

Output :-

```
Enter the no. of processes: 4
Enter the quantum:
4
Enter the process numbers:
1
2
3
4
Enter the Arrival time of processes:
1
2
3
4
Enter the Burst time of processes:
13
1
17
17
Process number Arrival time Burst time  Start time                  Final time      Wait Time       TurnAround Time
1              1            13          1 14 26 38                  39              25              38
2              2            1           5                           6               3               4
3              3            17          6 18 30 39 47               48              28              45
4              4            17          10 22 34 43 48               49             28              45
The average wait time is: 21
The average TurnAround time is: 33


...Program finished with exit code 0
Press ENTER to exit console.
```