PRACTICAL NO: 6

```
#include<stdio.h>
#include<stdlib.h>
int mutex=1,full=0,empty=3,x=0;
int main()
int n;
void producer();
void consumer();
int wait(int);
int signal(int);
printf("\n1.Producer\n2.Consumer\n3.Exit"
);
while(1)
printf("\nEnter your choice:");
scanf("%d",&n);
switch(n)
case 1: if((mutex==1)&&(empty!=0))
producer();
else
printf("Buffer is full!!\n");
break;
case 2: if((mutex==1)&&(full!=0))
consumer();
else
printf("Buffer is empty!!\n");
break;
case 3:
exit(0):
break;
return 0;
int wait(int s)
return (--s);
int signal(int s)
return(++s);
void producer()
mutex=wait(mutex);
```

```
full=signal(full);
empty=wait(empty);
x++;
printf("Producer produces the item
%d\n",x);
mutex=signal(mutex);
}
void consumer()
{
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);

printf("Consumer consumes item
%d\n",x);
x--;
mutex=signal(mutex);
}
```

Output:

```
C:\TURBOC3\Projects\sempophore.exe
1.Producer
2.Consumer
3.Exit
Enter your choice:1
Producer produces the item 1
Enter your choice:1
Producer produces the item 2
Enter your choice:1
Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:2
Consumer consumes item 3
Enter your choice:2
Consumer consumes item 2
Enter your choice:2
Consumer consumes item 1
Enter your choice:1
Producer produces the item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:1
Producer produces the item 1
Enter your choice:3
Process exited after 12.21 seconds with return value 0
Press any key to continue . . .
```

PRACTICAL NO: 5

Demonstrate the concept of non-preemptive scheduling algorithms.

1. FCFS

```
#include <stdio.h>
                                                         // Calculate waiting time for each process
                                                         for (int i = 0; i < N; i++)
// Function to Calculate waiting time,
                                                         {
// average waiting time, and average
                                                            wt[i] = tat[i] - bt[i];
turnaround time
                                                         }
void CalculateTimes(int at[], int bt[], int N)
{
                                                         // Print process details
  // Declare the arrays for waiting time,
                                                         printf("PN\tAT\tBT\tWT\tTAT\tCT\n\n");
  // turnaround time, and completion time
                                                         for (int i = 0; i < N; i++)
  int wt[N], tat[N], ct[N];
                                                         {
                                                            printf("%d\t%d\t%d\t%d\t%d\n", i +
  // Calculate completion time for each
                                                       1, at[i], bt[i], wt[i], tat[i], ct[i]);
process
                                                         }
  ct[0] = bt[0];
  for (int i = 1; i < N; i++)
                                                         // Calculate average waiting time and
                                                       average turnaround time
  {
     ct[i] = ct[i - 1] + bt[i];
                                                         float avg wt = 0.0, avg tat = 0.0; // Define
                                                       as float
  }
                                                         for (int i = 0; i < N; i++)
  // Calculate turnaround time for each process
                                                            avg_wt += wt[i];
  for (int i = 0; i < N; i++)
                                                            avg tat += tat[i];
  {
                                                         }
     tat[i] = ct[i] - at[i];
                                                         avg wt = N;
  }
                                                         avg tat = N;
```

```
}
  // Print average waiting time and average
                                                          printf("%d\n\n", total time);
turnaround time
                                                       }
  printf("\nAverage waiting time = \%.2f\n",
avg_wt);
                                                       // Driver code
  printf("Average turnaround time = \%.2f\n",
avg tat);
                                                       int main()
}
                                                       {
                                                         // Number of process
                                                          int N = 5;
// Function to print Gantt Chart
void PrintGanttChart(int at[], int bt[], int N)
                                                          // Array for Arrival time
{
  printf("\n\nGantt Chart:\n\n");
                                                          int at[] = \{0, 1, 2, 3, 4\};
                                                          // Array for Burst Time
  // Printing process numbers
  for (int i = 0; i < N; i++)
                                                          int bt[] = \{4, 3, 1, 2, 5\};
  {
     printf("| P%d ", i + 1);
                                                          // Function call to calculate times
  }
                                                          CalculateTimes(at, bt, N);
  printf("|\n");
                                                          // Function call to print Gantt Chart
  // Printing bars representing processes
                                                          PrintGanttChart(at, bt, N);
  int total time = 0;
  for (int i = 0; i < N; i++)
                                                          return 0;
     printf("%d ", total time);
     total time += bt[i];
```

Output:

```
Shaikhs-MacBook-Air:os aveis$ cd "/Users/aveis/Desktop/sem 4/
ac6.c -o os_prac6 && "/Users/aveis/Desktop/sem 4/practicals/d
                                   TAT
PN
        AT
                 BT
                          WT
                                           CT
1
        0
                 4
                                   4
                                           4
                          0
        1
                 3
                          3
                                   6
                                           7
2
        2
                          5
3
                 1
                                   6
                                           8
                          5
         3
                 2
4
                                            10
5
        4
                 5
                          6
                                   11
                                           15
Average waiting time = 3.80
Average turnaround time = 6.80
Gantt Chart:
| P1 | P2 | P3 | P4 | P5 |
         7
                 10
```

2. SJF

```
#include <stdio.h>
// Function to Calculate waiting time,
                                                                      int temp = bt copy[j];
// average waiting time, and average turnaround
                                                                      bt_{copy}[j] = bt_{copy}[j + 1];
time
                                                                      bt copy[j + 1] = temp;
void CalculateTimes(int at[], int bt[], int N)
                                                                      temp = process[j];
  // Declare the arrays for waiting time,
                                                                      process[j] = process[j + 1];
  // turnaround time, and completion time
                                                                      process[j + 1] = temp;
  int wt[N], tat[N], ct[N], bt copy[N], process[N];
                                                                    }
                                                                 }
  // Copy burst times to maintain the original array
                                                               }
  for (int i = 0; i < N; i++){
     bt copy[i] = bt[i];
                                                              // Calculate completion time for each process
     process[i] = i + 1;
                                                              ct[0] = bt copy[0];
  }
                                                               for (int i = 1; i < N; i++){
  // Sort processes based on burst time
                                                                 ct[i] = ct[i-1] + bt copy[i];
  for (int i = 0; i < N - 1; i++){
                                                               }
     for (int j = 0; j < N - i - 1; j++){
       if (bt copy[j] > bt <math>copy[j+1])
```

```
// Calculate turnaround time for each process
  for (int i = 0; i < N; i++){
                                                               // Printing process numbers
     tat[i] = ct[i] - at[process[i] - 1];
                                                               for (int i = 0; i < N; i++)
  }
                                                                 printf("| P%d ", i + 1);
                                                               }
  // Calculate waiting time for each process
  for (int i = 0; i < N; i++){
                                                               printf("|\n");
     wt[i] = tat[i] - bt[process[i] - 1];
  }
                                                               // Printing bars representing processes
  // Print process details
                                                               int total time = 0;
  printf("PN\tAT\tBT\tWT\tTAT\tCT\n\n");
                                                               for (int i = 0; i < N; i++)
  for (int i = 0; i < N; i++){
     printf("\%d\t\%d\t\%d\t\%d\t\%d\t\%d\n",
                                                                  printf("%d ", total time);
process[i], at[process[i] - 1], bt[process[i] - 1],
                                                                 total time += bt[i];
wt[i], tat[i], ct[i]);
                                                               }
  }
                                                               printf("%d\n\n", total_time);
  // Calculate average waiting time and average
turnaround time
  float avg wt = 0.0, avg tat = 0.0;
                                                            int main(){
  for (int i = 0; i < N; i++){
                                                               // Number of process
     avg wt += wt[i];
                                                               int N = 5;
     avg tat += tat[i];
  }
                                                               // Array for Arrival time
  avg wt = N;
                                                               int at[] = \{0, 1, 2, 3, 4\};
  avg tat = N;
                                                               // Array for Burst Time
  // Print average waiting time and average
                                                               int bt[] = \{4, 3, 1, 2, 5\};
turnaround time
  printf("\nAverage waiting time = \%.2f\n",
                                                               // Function call to calculate times
avg wt);
                                                               CalculateTimes(at, bt, N);
  printf("Average turnaround time = \%.2f\n",
avg tat);
                                                               // Function call to print Gantt Chart
                                                               PrintGanttChart(at, bt, N);
// Function to print Gantt Chart
                                                               return 0;
void PrintGanttChart(int at[], int bt[], int N){
                                                            }
  printf("\n\nGantt Chart:\n\n");
```

Output:

```
Shaikhs-MacBook-Air:os aveis$ cd "/Users/aveis/Desktop/sem 4/
 -o sjf && "/Users/aveis/Desktop/sem 4/practicals/os/"sjf
PN
                 BT
                          WT
                                  TAT
        AT
2
                                  0
                                           1
                          -1
4
        3
                 2
                                           3
                          -2
                                  0
5
                 5
                                           8
        4
                          -1
                                  4
1
                 6
                          8
                                           14
        0
                                  14
3
        2
                 8
                          12
                                           22
                                  20
Average waiting time = 3.20
Average turnaround time = 7.60
Gantt Chart:
  P1 | P2 | P3 | P4 | P5 |
    6
             15
                  17
                        22
```

3. Priority Scheduling

```
for (int i = 0; i < N; i++)
#include <stdio.h>
                                                           {
// Function to Calculate waiting time,
                                                             bt copy[i] = bt[i];
// average waiting time, and average
turnaround time
                                                             process[i] = i + 1;
void CalculateTimes(int at[], int bt[], int
                                                           }
priority[], int N)
                                                          // Sort processes based on priority
  // Declare the arrays for waiting time,
                                                           for (int i = 0; i < N - 1; i++)
  // turnaround time, and completion time
                                                           {
  int wt[N], tat[N], ct[N], bt_copy[N],
                                                             for (int i = 0; i < N - i - 1; i++)
process[N];
                                                             {
                                                                if (priority[j] > priority[j + 1] \parallel
  // Copy burst times to maintain the original
                                                        (priority[j] == priority[j+1] \&\& at[j] > at[j+1]
array
                                                        1]))
                                                                   int temp = bt copy[j];
```

```
bt copy[j] = bt copy[j + 1];
                                                       }
          bt copy[j + 1] = temp;
                                                       // Calculate waiting time for each process
                                                       for (int i = 0; i < N; i++)
         temp = process[j];
         process[j] = process[j + 1];
         process[j + 1] = temp;
                                                          wt[i] = tat[i] - bt[process[i] - 1];
                                                       }
          temp = at[j];
                                                       // Print process details
          at[j] = at[j + 1];
          at[j+1] = temp;
                                                     printf("PN\tAT\tBT\tPriority\tWT\tTAT\tCT\n\
                                                     n");
          temp = priority[j];
                                                       for (int i = 0; i < N; i++)
         priority[j] = priority[j + 1];
                                                       {
         priority[j + 1] = temp;
                                                     process[i], at[process[i] - 1], bt[process[i] - 1],
     }
                                                     priority[i], wt[i], tat[i], ct[i]);
  }
                                                       }
  // Calculate completion time for each
                                                       // Calculate average waiting time and
process
                                                     average turnaround time
  ct[0] = bt\_copy[0];
                                                       float avg_wt = 0.0, avg_tat = 0.0;
  for (int i = 1; i < N; i++)
                                                       for (int i = 0; i < N; i++)
  {
                                                       {
     ct[i] = ct[i-1] + bt copy[i];
                                                          avg wt += wt[i];
  }
                                                          avg_tat += tat[i];
                                                       }
  // Calculate turnaround time for each process
                                                       avg wt = N;
  for (int i = 0; i < N; i++)
                                                       avg tat = N;
     tat[i] = ct[i] - at[process[i] - 1];
```

```
// Print average waiting time and average
                                                       int main()
turnaround time
  printf("\nAverage waiting time = \%.2f\n",
                                                          // Number of process
avg_wt);
                                                          int N = 5;
  printf("Average turnaround time = \%.2f\n",
avg tat);
}
                                                          // Array for Arrival time
                                                          int at [] = {0, 1, 2, 3, 4};
// Function to print Gantt Chart
                                                          // Array for Burst Time
void PrintGanttChart(int at[], int bt[], int N)
{
                                                          int bt[] = \{4, 3, 1, 2, 5\};
  printf("\n\nGantt Chart:\n\n");
                                                          // Array for Priority
  // Printing process numbers
                                                          int priority[] = \{2, 1, 3, 4, 5\}; // Lower
                                                       number means higher priority
  for (int i = 0; i < N; i++)
                                                          // Function call to calculate times
     printf("| P\%d ", i + 1);
                                                          CalculateTimes(at, bt, priority, N);
  }
  printf("|\n");
                                                          // Function call to print Gantt Chart
                                                          PrintGanttChart(at, bt, N);
  // Printing bars representing processes
  int total time = 0;
                                                          return 0;
  for (int i = 0; i < N; i++)
     printf("%d ", total time);
     total time += bt[i];
  }
  printf("%d\n\n", total_time);
}
```

// Driver code

Output:

```
Shaikhs-MacBook-Air:os aveis$ cd "/Users/aveis/Desktop/sem 4/practicals/g
ity.c -o priority && "/Users/aveis/Desktop/sem 4/practicals/os/"priority
PN
        AT
                 BT
                         Priority
                                           WT
                                                   TAT
2
        0
                 3
                          1
                                           0
                                                   3
        1
                         2
                                           2
                                                   12
1
                 10
                                                            13
3
        2
                 8
                          3
                                           11
                                                   19
                                                            21
5
        3
                 5
                         4
                                           18
                                                   23
                                                            26
                 2
                         5
                                           22
                                                   24
                                                            28
Average waiting time = 10.60
Average turnaround time = 16.20
Gantt Chart:
 P1 | P2 | P3 | P4 | P5 |
    10
         13
```

4. Round Robin

```
}
#include <stdio.h>
// Function to Calculate waiting time,
                                                     // Initialize time and current completion
                                                  time
// average waiting time, and average
turnaround time
                                                     int time = 0, current ct = 0;
void CalculateTimes(int at[], int bt[], int
quantum, int N)
                                                     // Process until all processes are done
                                                     while (1)
  // Declare the arrays for waiting time,
  // turnaround time, and completion time
                                                       int done = 1; // Assume all processes
  int wt[N], tat[N], ct[N],
                                                  are done initially
remaining bt[N];
                                                       // Traverse all processes
  // Initialize remaining burst time
                                                        for (int i = 0; i < N; i++)
  for (int i = 0; i < N; i++)
                                                        {
   {
                                                          // If burst time is remaining for this
     remaining bt[i] = bt[i];
                                                  process
```

```
{
                                                          break;
          done = 0; // There are still
                                                     }
processes remaining
                                                    // Calculate turnaround time for each
          // Execute this process for a
                                                  process
quantum or remaining time, whichever is
                                                    for (int i = 0; i < N; i++)
smaller
                                                     {
          if (remaining bt[i] > quantum)
                                                       tat[i] = ct[i] - at[i];
                                                     }
            // Update time and remaining
burst time
                                                    // Calculate waiting time for each process
            time += quantum;
                                                    for (int i = 0; i < N; i++)
            remaining bt[i] -= quantum;
                                                     {
          }
                                                       wt[i] = tat[i] - bt[i];
          else
                                                     }
            // Update time and remaining
burst time
                                                    // Print process details
            time += remaining bt[i];
            remaining bt[i] = 0;
                                                  printf("PN\tAT\tBT\tWT\tTAT\tCT\n\n");
                                                    for (int i = 0; i < N; i++)
                                                     {
            // Calculate completion time
for this process
                                                       printf("%d\t%d\t%d\t%d\t%d\t%d\n",
                                                  i + 1, at[i], bt[i], wt[i], tat[i], ct[i]);
            ct[i] = time;
                                                     }
          }
       }
                                                    // Calculate average waiting time and
     }
                                                  average turnaround time
                                                    float avg wt = 0.0, avg tat = 0.0;
     // If all processes are done, exit the
                                                    for (int i = 0; i < N; i++)
loop
```

```
// Number of process
                                                    int N = 5;
     avg_wt += wt[i];
     avg tat += tat[i];
  }
                                                    // Array for Arrival time
                                                    int at[] = \{0, 1, 2, 3, 4\};
  avg wt = N;
  avg tat = N;
                                                    // Array for Burst Time
  // Print average waiting time and average
                                                    int bt[] = \{4, 3, 1, 2, 5\};
turnaround time
  printf("\nAverage waiting time =
                                                    // Time quantum
%.2f\n", avg wt);
                                                    int quantum = 2;
  printf("Average turnaround time =
%.2f\n'', avg tat);
                                                    // Function call to calculate times
}
                                                    CalculateTimes(at, bt, quantum, N);
int main()
                                                    return 0;
{
                                                  }
```

Output: