**LAB ASSIGNMENT - 5**

**Scheduling algorithms**

**1. First Come First Serve (FCFS)**

C Code-

| #include <stdio.h>  int main() {  int n, i;  printf("Enter number of processes: ");  scanf("%d", &n);    int bt[n], wt[n], tat[n];  printf("Enter burst times:\n");  for(i = 0; i < n; i++) {  printf("P%d: ", i+1);  scanf("%d", &bt[i]);  }   wt[0] = 0;  for(i = 1; i < n; i++)  wt[i] = wt[i - 1] + bt[i - 1];   for(i = 0; i < n; i++)  tat[i] = wt[i] + bt[i];   printf("\nProcess\tBT\tWT\tTAT\n");  for(i = 0; i < n; i++)  printf("P%d\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);   return 0; } |
| --- |

Output -

| Enter number of processes: 3 Enter burst times: P1: 5 P2: 8 P3: 6  Process BT WT TAT P1 5 0 5 P2 8 5 13 P3 6 13 19 |
| --- |

**2. Shortest Job First (SJF)**

C Code -

| #include <stdio.h>  int main() {  int n, i, j;  printf("Enter number of processes: ");  scanf("%d", &n);    int bt[n], p[n], wt[n], tat[n], temp;  printf("Enter burst times:\n");  for(i = 0; i < n; i++) {  printf("P%d: ", i+1);  scanf("%d", &bt[i]);  p[i] = i+1;  }   *// Sort processes by burst time*  for(i = 0; i < n - 1; i++) {  for(j = i + 1; j < n; j++) {  if(bt[i] > bt[j]) {  temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;  temp = p[i]; p[i] = p[j]; p[j] = temp;  }  }  }   wt[0] = 0;  for(i = 1; i < n; i++)  wt[i] = wt[i-1] + bt[i-1];   for(i = 0; i < n; i++)  tat[i] = wt[i] + bt[i];   printf("\nProcess\tBT\tWT\tTAT\n");  for(i = 0; i < n; i++)  printf("P%d\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);   return 0; } |
| --- |

Output -

| Enter number of processes: 3 Enter burst times: P1: 8 P2: 4 P3: 2  Process BT WT TAT P3 2 0 2 P2 4 2 6 P1 8 6 14 |
| --- |

**3. Round Robin Scheduling**

C Code -

| #include <stdio.h>  int main() {  int i, n, time = 0, remain, tq;  printf("Enter number of processes: ");  scanf("%d", &n);   int bt[n], rt[n], wt[n], tat[n];  remain = n;   printf("Enter burst times:\n");  for(i = 0; i < n; i++) {  printf("P%d: ", i+1);  scanf("%d", &bt[i]);  rt[i] = bt[i];  }   printf("Enter time quantum: ");  scanf("%d", &tq);   int done;  do {  done = 1;  for(i = 0; i < n; i++) {  if(rt[i] > 0) {  done = 0;  if(rt[i] > tq) {  time += tq;  rt[i] -= tq;  } else {  time += rt[i];  tat[i] = time;  wt[i] = tat[i] - bt[i];  rt[i] = 0;  }  }  }  } while(!done);   printf("\nProcess\tBT\tWT\tTAT\n");  for(i = 0; i < n; i++)  printf("P%d\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);   return 0; } |
| --- |

Output -

| Enter number of processes: 3 Enter burst times: P1: 10 P2: 5 P3: 8 Enter time quantum: 4  Process BT WT TAT P1 10 9 19 P2 5 4 9 P3 8 9 17 |
| --- |