EXPERIMENT NO:6 b

SCHEDULING ALGORITHMS

AIM

To implement different scheduling algorithms.

B. SJF (Shortest Job First)

ALGORITHM

Step 1: Start the program.
Step 2: Input the number of processes (n).
Step 3: Input the Burst Time (BT) for each process.

- Assign a unique Process ID (**P[i]**) for identification.

Step 4: Sort the processes based on Burst Time in ascending order (Shortest Job First).

- **For** each process `i` from `0` to `n-1`:
  - **For** each process `j` from `i+1` to `n-1`:
    - **If** `BT[i] > BT[j]`:
      - **Swap** `BT[i]` with `BT[j]`.
      - **Swap** the corresponding Process IDs `P[i]` and `P[j]`.

Step 5: Initialize Turnaround Time (TAT) and Waiting Time (WT) arrays.
Step 6: Calculate Turnaround Time (TAT) for each process.

- **TAT[0] = BT[0]** (The first process's turnaround time is equal to its burst time).
- **For** each subsequent process `i` from `1` to `n-1`:
  - **TAT[i] = TAT[i-1] + BT[i]** (Current TAT = Previous TAT + Current Burst Time).
- **Compute Total Turnaround Time:**
  - `Total_TAT = Σ TAT[i]` (Sum of all turnaround times).
- **Compute Average Turnaround Time:**
  - `Avg_TAT = Total_TAT / n`

Step 7: Calculate Waiting Time (WT) for each process.

- **WT[0] = 0** (The first process has no waiting time).
- **For** each subsequent process `i` from `1` to `n-1`:
  - **WT[i] = WT[i-1] + BT[i-1]** (Current WT = Previous WT + Previous Burst Time).
- **Compute Total Waiting Time:**
  - `Total_WT = Σ WT[i]` (Sum of all waiting times).
- **Compute Average Waiting Time:**
  - `Avg_WT = Total_WT / n`

Step 8: Display the following for each process:

- **Process ID**, **Burst Time**, **Turnaround Time (TAT)**, and **Waiting Time (WT)**.

Step 9: Display the Total and Average Turnaround Time and Waiting Time.
Step 10: Stop the program.

PROGRAM

```
#include <stdio.h>

void swap(int *a, int *b);

int main() {
   int i, j, n, tot_tat = 0, tot_wt = 0;
   int p[30], bt[30], tat[30], wt[30];
   float avg_tat, avg_wt;

   printf("Enter the no.of processes: ");
   scanf("%d", &n);

   printf("Enter burst time for each process:\n");
   for(i = 0; i < n; i++) {
      scanf("%d", &bt[i]);
      p[i] = i;
   }

                                          // Sorting for SJF
   for(i = 0; i < n; i++) {
      for(j = i + 1; j < n; j++) {
         if(bt[i] > bt[j]) {
            swap(&bt[i], &bt[j]);
            swap(&p[i], &p[j]);
         }
      }
   }

                                          // Turnaround Time
   for(i = 0; i < n; i++) {
      tat[i] = (i == 0) ? bt[i] : tat[i - 1] + bt[i];
      tot_tat += tat[i];
   }

                                          // Waiting Time
   wt[0] = 0;
   for(i = 1; i < n; i++) {
      wt[i] = wt[i - 1] + bt[i - 1];
      tot_wt += wt[i];
   }

                                          // Output
   printf("\nPROCESS\tBURST TIME\tTURNAROUND TIME\tWAITING TIME\n");
   for(i = 0; i < n; i++)
```

```
        printf("P[%d]\t\t%d\t\t%d\t\t%d\n", p[i] + 1, bt[i], tat[i], wt[i]);

    avg_tat = (float)tot_tat / n;
    avg_wt = (float)tot_wt / n;

    printf("\nTotal Turnaround Time: %d", tot_tat);
    printf("\nAverage Turnaround Time: %.2f", avg_tat);
    printf("\nTotal Waiting Time: %d", tot_wt);
    printf("\nAverage Waiting Time: %.2f\n", avg_wt);

    return 0;
}

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

OUTPUT

```
Enter the no.of processes
4
Enter burst time for each process
8
5
4
7
```

PROCESS	BURST TIME	TURN AROUND TIME	WAITING TIME
process[3]	4	4	0
process[2]	5	9	4
process[4]	7	16	9
process[1]	8	24	16

```
Total Turn around Time:53
Average Turn around Time :13
Total Waiting Time:29
Total avg. Waiting Time:7
```