B.BHANUTEJA REDDY-192325016

4. Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.

Aim:

To design a CPU scheduling program using the Shortest Job Next (SJN) or Shortest Job First (SJF) technique, which selects the waiting process with the smallest execution time to execute next.

Aim:

To design a CPU scheduling program using the Shortest Job Next (SJN) or Shortest Job First (SJF) technique, which selects the waiting process with the smallest execution time to execute next.

Algorithm:

1. Start the program.
2. Input the number of processes and their burst times.
3. Sort the processes based on their burst times in ascending order.
4. Calculate the waiting time for each process:
   - Waiting time for the first process is 0.
   - For subsequent processes, Waiting Time[i] = Waiting Time[i-1] + Burst Time[i-1].
5. Calculate the turnaround time for each process:
   - Turnaround Time[i] = Waiting Time[i] + Burst Time[i].
6. Display the process details, including their burst time, waiting time, and turnaround time.
7. Compute the average waiting time and turnaround time.
8. End the program.

Procedure:

1. Include necessary headers: <stdio.h>.
2. Define arrays for process IDs, burst times, waiting times, and turnaround times.

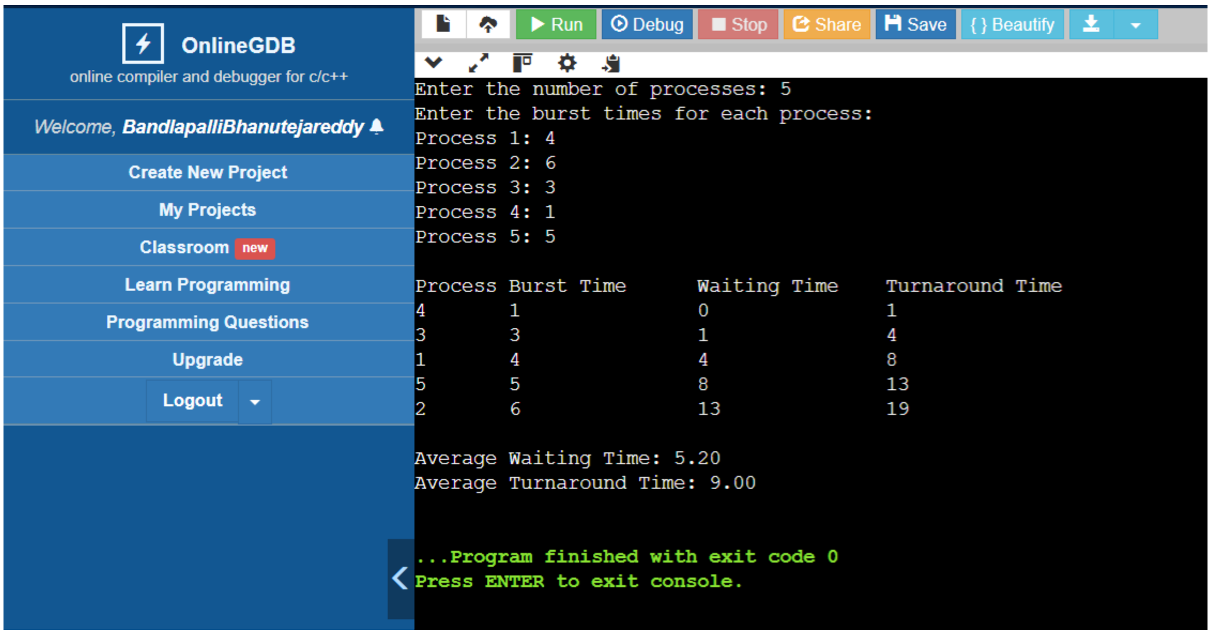
3. Sort the processes by burst time using a simple sorting algorithm (e.g., Bubble Sort).
4. Compute waiting times and turnaround times iteratively.
5. Calculate and display average waiting and turnaround times.

CODE:

```c
#include <stdio.h>


int main() {
   int n, i, j, temp;
   float avg_wait = 0, avg_turnaround = 0;


   printf("Enter the number of processes: ");
   scanf("%d", &n);


   int process[n], burst_time[n], waiting_time[n], turnaround_time[n];


   printf("Enter the burst times for each process:\n");
   for (i = 0; i < n; i++) {
      process[i] = i + 1;
      printf("Process %d: ", i + 1);
      scanf("%d", &burst_time[i]);
   }


   for (i = 0; i < n - 1; i++) {
      for (j = 0; j < n - i - 1; j++) {
         if (burst_time[j] > burst_time[j + 1]) {
            temp = burst_time[j];
            burst_time[j] = burst_time[j + 1];
            burst_time[j + 1] = temp;


            temp = process[j];
            process[j] = process[j + 1];
            process[j + 1] = temp;
         }
      }
   }

   waiting_time[0] = 0;
   for (i = 1; i < n; i++) {
      waiting_time[i] = waiting_time[i - 1] + burst_time[i - 1];
   }

   for (i = 0; i < n; i++) {
      turnaround_time[i] = waiting_time[i] + burst_time[i];
      avg_wait += waiting_time[i];
      avg_turnaround += turnaround_time[i];
   }

   avg_wait /= n;
   avg_turnaround /= n;

   printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
   for (i = 0; i < n; i++) {
      printf("%d\t%d\t\t%d\t\t%d\n", process[i], burst_time[i], waiting_time[i],
turnaround_time[i]);
   }
```

```
    printf("\nAverage Waiting Time: %.2f\n", avg_wait);
    printf("Average Turnaround Time: %.2f\n", avg_turnaround);

    return 0;
}
```

OUTPUT:



```
Enter the number of processes: 5
Enter the burst times for each process:
Process 1: 4
Process 2: 6
Process 3: 3
Process 4: 1
Process 5: 5

Process Burst Time      Waiting Time    Turnaround Time
4       1               0               1
3       3               1               4
1       4               4               8
5       5               8               13
2       6               13              19

Average Waiting Time: 5.20
Average Turnaround Time: 9.00


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