2) Preemptive Mode

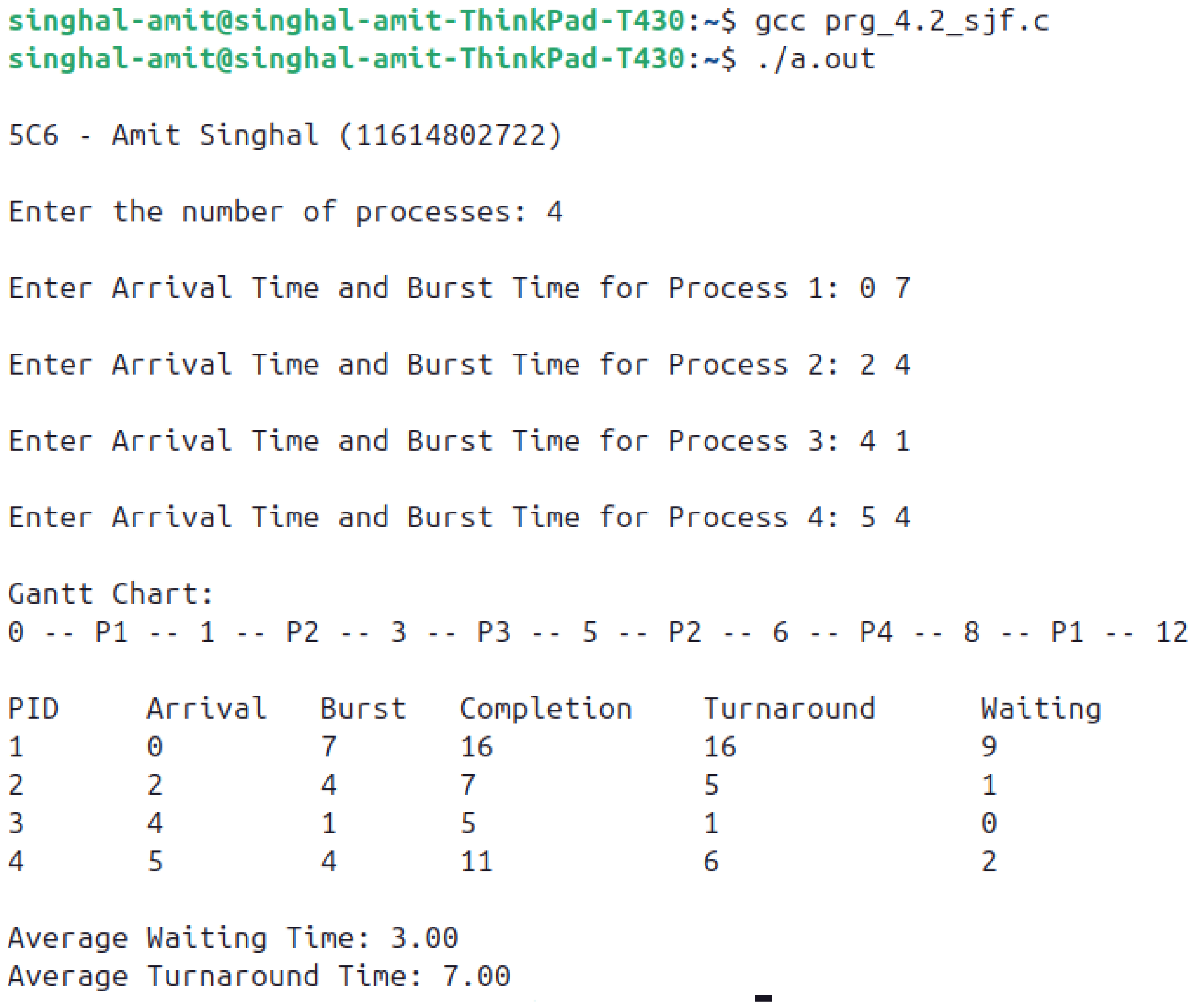
Source_Code ::

```
#include <stdio.h>

typedef struct
{
  int pid;        // Process ID
  int arrival;    // Arrival time
  int burst;      // Burst time
  int remaining;  // Remaining burst time (for preemption)
  int completion; // Completion time
  int waiting;    // Waiting time
  int turnaround; // Turnaround time
} Process;

// Function to find the process with the shortest remaining time at a given time
int findShortestRemaining(Process *p, int n, int time)
{
  int min_index = -1;
  int min_remaining = 99999;

  for (int i = 0; i < n; i++)
  {
    if (p[i].arrival <= time && p[i].remaining > 0 && p[i].remaining < min_remaining)
    {
      min_remaining = p[i].remaining;
      min_index = i;
    }
  }

  return min_index;
}

void sjfPreemptive(Process *p, int n)
{
  int time = 0;       // Current time
  int completed = 0; // Number of completed processes
  int gantt[100];    // Gantt chart sequence
  int gantt_index = 0;

  while (completed < n)
  {
    int shortest_job = findShortestRemaining(p, n, time);

    if (shortest_job == -1)
    {
      // If no process is ready, increment the time (idle)
      time++;
      gantt[gantt_index++] = -1;
    }
    else
    {
      // Execute the process for 1 unit of time
      p[shortest_job].remaining--;
      gantt[gantt_index++] = shortest_job;

      time++;

      // If the process is finished
      if (p[shortest_job].remaining == 0)
      {
        p[shortest_job].completion = time;
        p[shortest_job].turnaround = p[shortest_job].completion - p[shortest_job].arrival;
        p[shortest_job].waiting = p[shortest_job].turnaround - p[shortest_job].burst;
        completed++;
      }
    }
  }

  // Gantt chart display
  printf("\nGantt Chart:\n");
  printf("0"); // Start at time 0
  int current_time = 0;
  for (int i = 0; i < gantt_index; i++)
  {
    if (gantt[i] == -1)
    {
      printf(" -- XX -- %d", ++current_time); // Idle time
    }
    else
    {
      if (i == 0 || gantt[i] != gantt[i - 1])
      { // Only display if process changes
        printf(" -- P%d -- %d", p[gantt[i]].pid, ++current_time);
      }
      else
      {
        current_time++;
      }
    }
  }
  printf("\n");
}

// Function to display the process table
void displayResults(Process *p, int n)
{
  printf("\nPID\tArrival\t  Burst\t  Completion\tTurnaround\tWaiting\n");
  for (int i = 0; i < n; i++)
  {
    printf("%d\t%d\t  %d\t  %d\t\t%d\t\t%d\n", p[i].pid, p[i].arrival,
p[i].burst,
           p[i].completion, p[i].turnaround, p[i].waiting);
  }
}

// Function to calculate and display average times
void calculateAverages(Process *p, int n)
```

```
{
  float total_waiting = 0, total_turnaround = 0;

  for (int i = 0; i < n; i++)
  {
    total_waiting += p[i].waiting;
    total_turnaround += p[i].turnaround;
  }

  printf("\nAverage Waiting Time: %.2f", total_waiting / n);
  printf("\nAverage Turnaround Time: %.2f\n", total_turnaround / n);
}

int main()
{
  int n;
  printf("\n5C6 - Amit Singhal (11614802722)\n");
  printf("\nEnter the number of processes: ");
  scanf("%d", &n);

  Process p[n];

  // Input the arrival and burst times for each process
  for (int i = 0; i < n; i++)
  {
    p[i].pid = i + 1;
    printf("\nEnter Arrival Time and Burst Time for Process %d: ", i + 1);
    scanf("%d%d", &p[i].arrival, &p[i].burst);
```

```
      p[i].remaining = p[i].burst; // Remaining burst time for preemption
      p[i].completion = 0;        // Initially no completion time
    }

    sjfPreemptive(p, n);
    displayResults(p, n);
    calculateAverages(p, n);

    return 0;
  }
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~$ gcc prg_4.2_sjf.c
singhal-amit@singhal-amit-ThinkPad-T430:~$ ./a.out

5C6 - Amit Singhal (11614802722)

Enter the number of processes: 4

Enter Arrival Time and Burst Time for Process 1: 0 7

Enter Arrival Time and Burst Time for Process 2: 2 4

Enter Arrival Time and Burst Time for Process 3: 4 1

Enter Arrival Time and Burst Time for Process 4: 5 4

Gantt Chart:
0 -- P1 -- 1 -- P2 -- 3 -- P3 -- 5 -- P2 -- 6 -- P4 -- 8 -- P1 -- 12

PID       Arrival    Burst    Completion    Turnaround       Waiting
1         0          7        16            16               9
2         2          4        7             5                1
3         4          1        5             1                0
4         5          4        11            6                2

Average Waiting Time: 3.00
Average Turnaround Time: 7.00
```