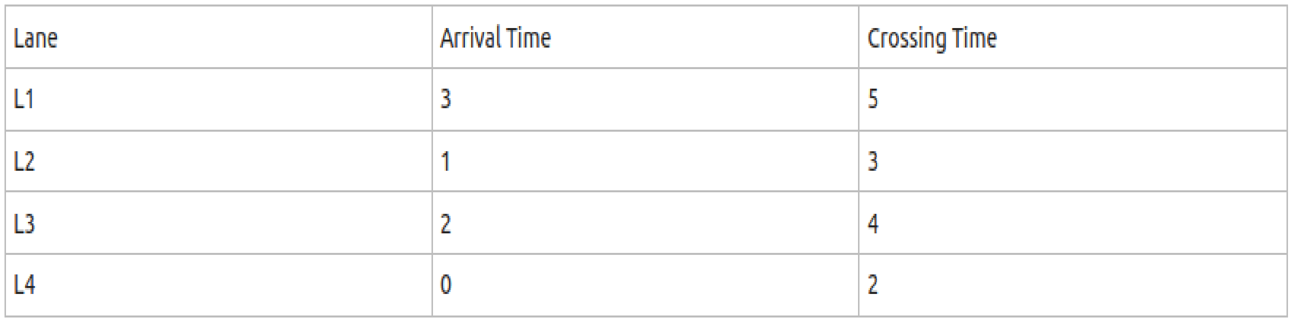
0S LAB 7

21BAI1217

MAINAK CHATTOPADHYAY

1. Consider a junction of roads where 4 lanes meet at this junction. Everytime, there is a frequent problem of jamming of vehicles at this junction. To overcome the problem, traffic signal was planned to be installed at the junction. The operation of the signal is planned in such a way that vehicles on each side will be allowed only for 3 minutes and then it will be stopped. The chance will be given to the next side of the vehicles. Write a program in C to implement an appropriate scheduling algorithm to illustrate how the vehicles cross the junction without jamming. Calculate the average waiting time for the vehicles on each lane. Display the results on the screen.



Lane	Arrival Time	Crossing Time
L1	3	5
L2	1	3
L3	2	4
L4	0	2

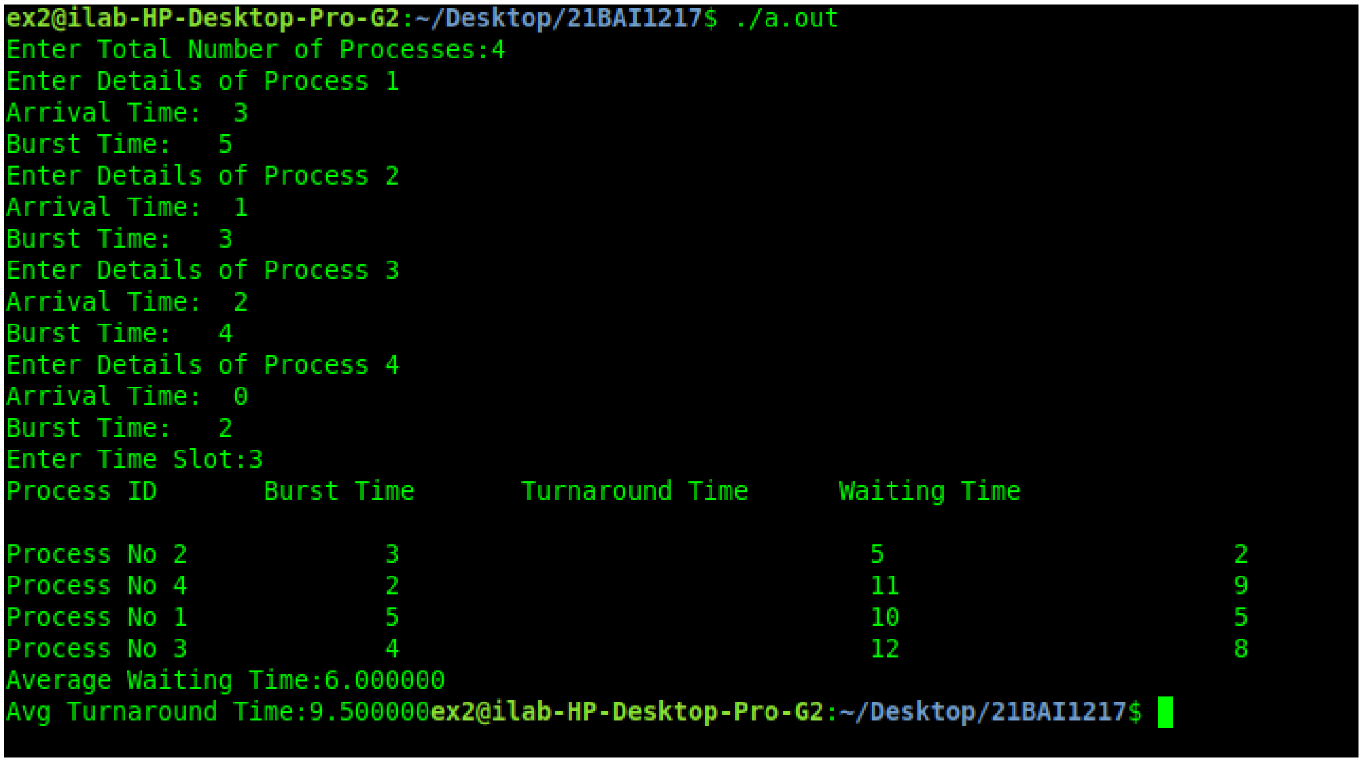
CODE

```
#include<stdio.h>

int main()
{
   //Input no of processed
   int  n;
   printf("Enter Total Number of Processes:");
   scanf("%d", &n);
   int wait_time = 0, ta_time = 0, arr_time[n], burst_time[n], temp_burst_time[n];
   int x = n;

   //Input details of processes
   for(int i = 0; i < n; i++)
   {
      printf("Enter Details of Process %d \n", i + 1);
      printf("Arrival Time:  ");
      scanf("%d", &arr_time[i]);
      printf("Burst Time:   ");
      scanf("%d", &burst_time[i]);
      temp_burst_time[i] = burst_time[i];
   }

   //Input time slot
```

```
int time_slot;
printf("Enter Time Slot:");
scanf("%d", &time_slot);

//Total indicates total time
//counter indicates which process is executed
int total = 0,  counter = 0,i;
printf("Process ID       Burst Time       Turnaround Time      Waiting Time\n");
for(total=0, i = 0; x!=0; )
{
   // define the conditions
   if(temp_burst_time[i] <= time_slot && temp_burst_time[i] > 0)
   {
      total = total + temp_burst_time[i];
      temp_burst_time[i] = 0;
      counter=1;
   }
   else if(temp_burst_time[i] > 0)
   {
      temp_burst_time[i] = temp_burst_time[i] - time_slot;
      total  += time_slot;
   }
   if(temp_burst_time[i]==0 && counter==1)
   {
      x--; //decrement the process no.
      printf("\nProcess No %d  \t\t %d\t\t\t\t %d\t\t\t %d", i+1, burst_time[i],
           total-arr_time[i], total-arr_time[i]-burst_time[i]);
      wait_time = wait_time+total-arr_time[i]-burst_time[i];
      ta_time += total -arr_time[i];
      counter =0;
   }
   if(i==n-1)
   {
      i=0;
   }
   else if(arr_time[i+1]<=total)
   {
      i++;
   }
   else
   {
      i=0;
   }
}
```

```
    float average_wait_time = wait_time * 1.0 / n;
    float average_turnaround_time = ta_time * 1.0 / n;
    printf("\nAverage Waiting Time:%f", average_wait_time);
    printf("\nAvg Turnaround Time:%f", average_turnaround_time);
    return 0;
}
```

<u>OUTPUT</u>



```
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BAI1217$ ./a.out
Enter Total Number of Processes:4
Enter Details of Process 1
Arrival Time:  3
Burst Time:   5
Enter Details of Process 2
Arrival Time:  1
Burst Time:   3
Enter Details of Process 3
Arrival Time:  2
Burst Time:   4
Enter Details of Process 4
Arrival Time:  0
Burst Time:   2
Enter Time Slot:3
Process ID        Burst Time        Turnaround Time       Waiting Time

Process No 2            3                             5                       2
Process No 4            2                             11                      9
Process No 1            5                             10                      5
Process No 3            4                             12                      8
Average Waiting Time:6.000000
Avg Turnaround Time:9.500000ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BAI1217$ 
```

2.Write a C program to show how a child process becomes zombie process.

CODE-

```
// A C program to demonstrate Zombie Process.
// Child becomes Zombie as parent is sleeping
// when child process exits.
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
        // Fork returns process id
        // in parent process
        pid_t child_pid = fork();

        // Parent process
        if (child_pid > 0)
        {
                sleep(10);
        }
        // Child process
        else
        {
                exit(0);
        }
        return 0;
}
```

OUTPUT

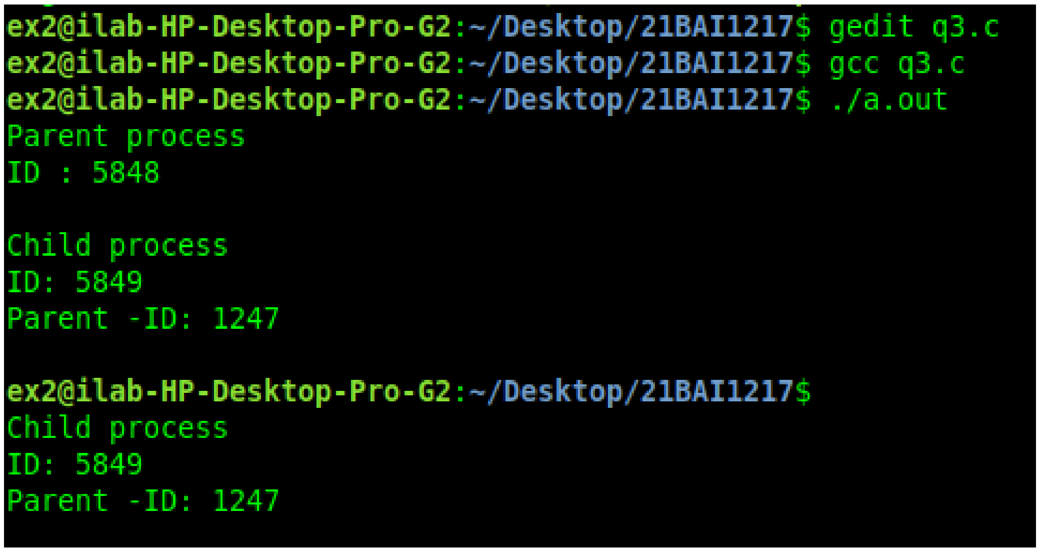
Then command prompt will wait for some time(60 sec) and then again command prompt will appear later.

3. Write a C program to show when a child process becomes an orphan process.

CODE -

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
   // fork() Create a child process

   int pid = fork();
   if (pid > 0) {
      //getpid() returns process id
      // while getppid() will return parent process id
      printf("Parent process\n");
      printf("ID : %d\n\n", getpid());
   }
   else if (pid == 0) {
      printf("Child process\n");
      // getpid() will return process id of child process
      printf("ID: %d\n", getpid());
      // getppid() will return parent process id of child process
      printf("Parent -ID: %d\n\n", getppid());

      sleep(10);

      // At this time parent process has finished.
      // So if u will check parent process id
      // it will show different process id
      printf("\nChild process \n");
      printf("ID: %d\n", getpid());
      printf("Parent -ID: %d\n", getppid());
   }
   else {
      printf("Failed to create child process");
   }

   return 0;
}
```

OUTPUT-



```
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BAI1217$ gedit q3.c
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BAI1217$ gcc q3.c
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BAI1217$ ./a.out
Parent process
ID : 5848

Child process
ID: 5849
Parent -ID: 1247

ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BAI1217$
Child process
ID: 5849
Parent -ID: 1247
```