Student Name : Siddhant Jain

Section : K1618, 2

Roll no : 46

Registration no: 11606071

Email Address: jainsid987@gmail.com

Githublink: https://github.com/11606071/OS-SIDDHANT-B46

1. Explain the problem in terms of operating system concept? (Max 200 word

Ans.. In this we implement Shortest job with preemptive method.In Shortest job we implement that process among the processes which are arrived at the same time which having the shortest burst time..and in preemptive we switch another processs..for example if one process is running at the mean another process arrives then our prpcessor checks which process having the shortest burst time ..if new process that arrives having shortest bust time than current5ly running process than we switch..

2. Write the algorithm for proposed solution of the assigned problem

Ans.. Step 1: Create the linked list having three parameters in it i.e

- Burst time of the process
- Arrival time of the process
- Next part which contain the address the next node

Step 2: Take the input how many processes are going to run …and make linked list according to that and store the inputs in the nodes and store NULL in the laast node next part.

Step 3: Sort the linked list by selection sort according to the arrival time store in the nodes.. and do changes in the next part of the node according to that.

Step 4; In while loop we pass the condition whether our variable value is less than or equal to CPU running time until our while loop would be runned.

Step 4: WE store the node address in another pointer type variable and store current6 node address in another pointer type variable..t

Step 5: then we pass the condition until our next node arrives our current burst would decrement by 1 and varaiable which checjs CPU running time would increment by 1

Step 6: If a condition is arises i.e if currently running process burst is not equal to zero and another process arrives then we check the process which having the shortest burst time and start executing that process

Step 7: by this we calculate Turn around time and waiting time for every process

3. Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

struct node{

int arrival;

int burst;

int totalBurst;

struct node *next;

}*temp,*ptr,*p,*save,*q,*g,*o,*y,*parent,*c;
```

```
int main()
{
   struct node *temp;                                                    //O(1)
int i,e,n,d,k=0,ft=0,fw=0;
//O(1)
printf("enter the no of processes:::");                                  //O(1)
scanf("%d",&n);                                                          //O(1)
o=ptr=q=g=temp=(struct node*)malloc(sizeof(struct node));                //O(1)
printf("enter the arrival time");                                        //O(1)
scanf("%d",&temp->arrival);                                              //O(1)
printf("enter the burst time");                                          //O(1)
scanf("%d",&temp->burst);                                                //O(1)
temp->totalBurst=temp->burst;                                            //O(1)
temp->next=NULL;                                                         //O(1)
for(i=1;i<n;i++)                                                         //O(n)
{
temp->next=(struct node*)malloc(sizeof(struct node));                    //O(n)
temp=temp->next;                                                         //O(n)
printf("enter the arrival time");                                        //O(n)
scanf("%d",&temp->arrival);                                              //O(n)
printf("enter the burst time");                                          //O(n)
scanf("%d",&temp->burst);                                                //O(n)
temp->totalBurst=temp->burst;                                            //O(n)
```

```
temp->next=NULL;                                                    //O(n)
}
for(i=0;i<n;i++)
{
p=ptr;                                                              //O(n)
while(save!=NULL)
{
p=ptr;                                                  //O(n*n)
if(p->arrival>save->arrival)                            //O(n*n)
{
e=save->arrival;                                        //O(n*n)
save->arrival=p->arrival;                               //O(n*n)
p->arrival=e;                                           //O(n*n)
e=save->burst;                                          //O(n*n)
save->burst=p->burst;                                   //O(n*n)
p->burst=e;                                             //O(n*n)
p->totalBurst=p->burst;                                 //O(n*n)
save->totalBurst=save->burst;                           //O(n*n)
}
save=save->next;                                        //O(n*n)
}
ptr=ptr->next;                                          //O(n)
}
```

```
q=g;                                                    //O(1)
k=q->arrival;                                           //O(1)
int r=0;                                                //O(1)
while(k<200)
{
   parent=q->next;                                      //O(n)
   c=q;                                                 //O(n)
   if(parent!=NULL)
   {
   while(parent->arrival<=k&&r!=1)                      //O(n)
   {
   if(parent->burst<q->burst)                           //O(n*n)
   {
      c=parent;                                         //O(n*n
   }
   parent=parent->next;                                 //O(n*n)
   if(parent==NULL)                                     //O(n*n)
   {
      r=1;                                              //O(n*n)
      break;                                            //O(n*n)
   }
   }
   }
```

```
c->burst--;                                                    //O(n)
if(c->burst==0)
{
   printf("Process executed at:%d\n",k+1);                     //O(n)
ft+=k-c->arrival+1;
printf("Turn Around time of Process:%d\n",k-c->arrival+1);
fw+=k-c->arrival-c->totalBurst + 1;  //O(n)
printf("Waiting Time:%d\n",k-c->arrival-c->totalBurst+1);     //O(n)
if(o==c)
{
   q=q->next;                                                  //O(n)
   g=q;                                                        //O(n)
   o=q;                                                        //O(n)
}
else
{
while(o!=c)                                                    //O(n)
{
    y=o;                                                       //O(n*n)
    o=o->next;                                                 //O(n*n)
}
 y->next=y->next->next;                                        //O(n)
}
```

```
printf("\n");                                                    //O(n)
}
k++;                                                             //      O(n)
}
printf("avg.waiting  time  :%d",ceil(fw/n));
printf("avg.turnaroundtime:%d",ceil(fw/n));
}
```

The complexity of the program is O(n*n)

4. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

The program uses Linked Lists and so there is no limit as to how many processes can be added to the program for execution. The only constraint in this code is the amount of CPU running time which will be entered by the user. The total burst time and IDLE time can not exceed to CPU running time.

5 Explain all the test cases applied on the solution of assigned problem

S.NO	Arrival time	Burst time	Turnaround time	Waiting time
1	2	5	6	1
2	4	6	10	4
3	1	2	2	0

In this problem we take a test case having different arrival time and different burst time .we solve

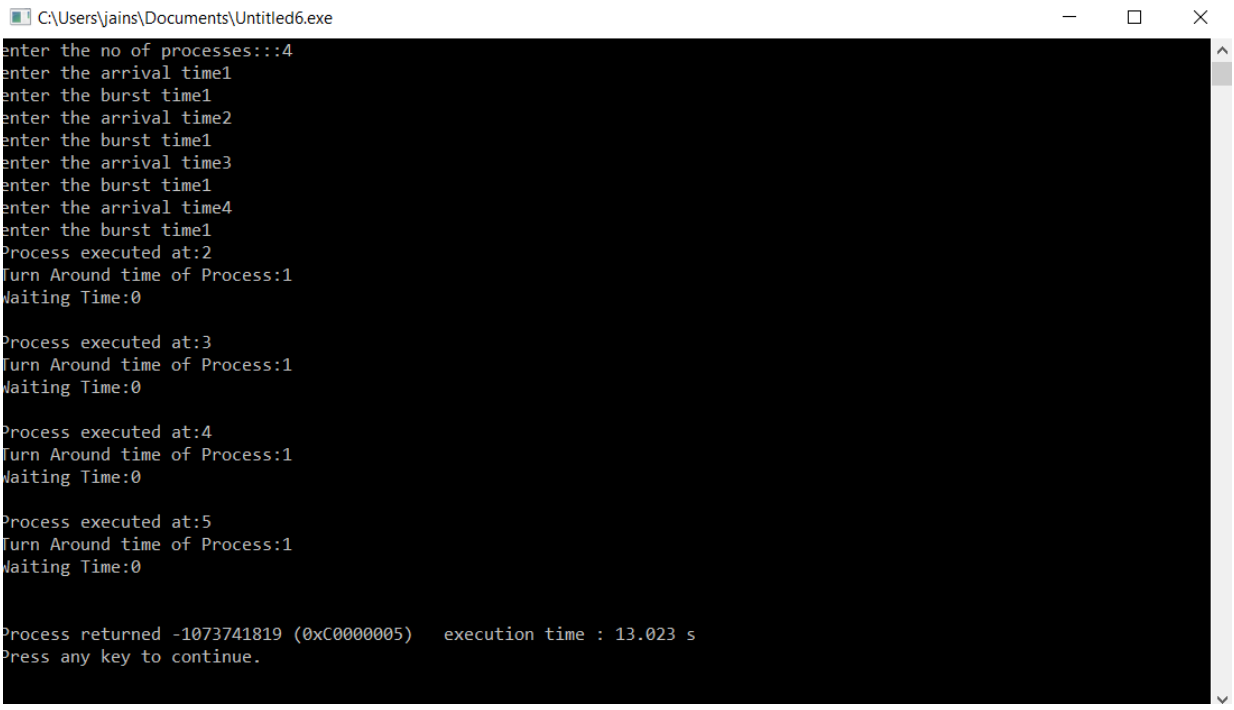
It by shortest job first by using pre-emption the process having leastr burst time would execute first..we find..

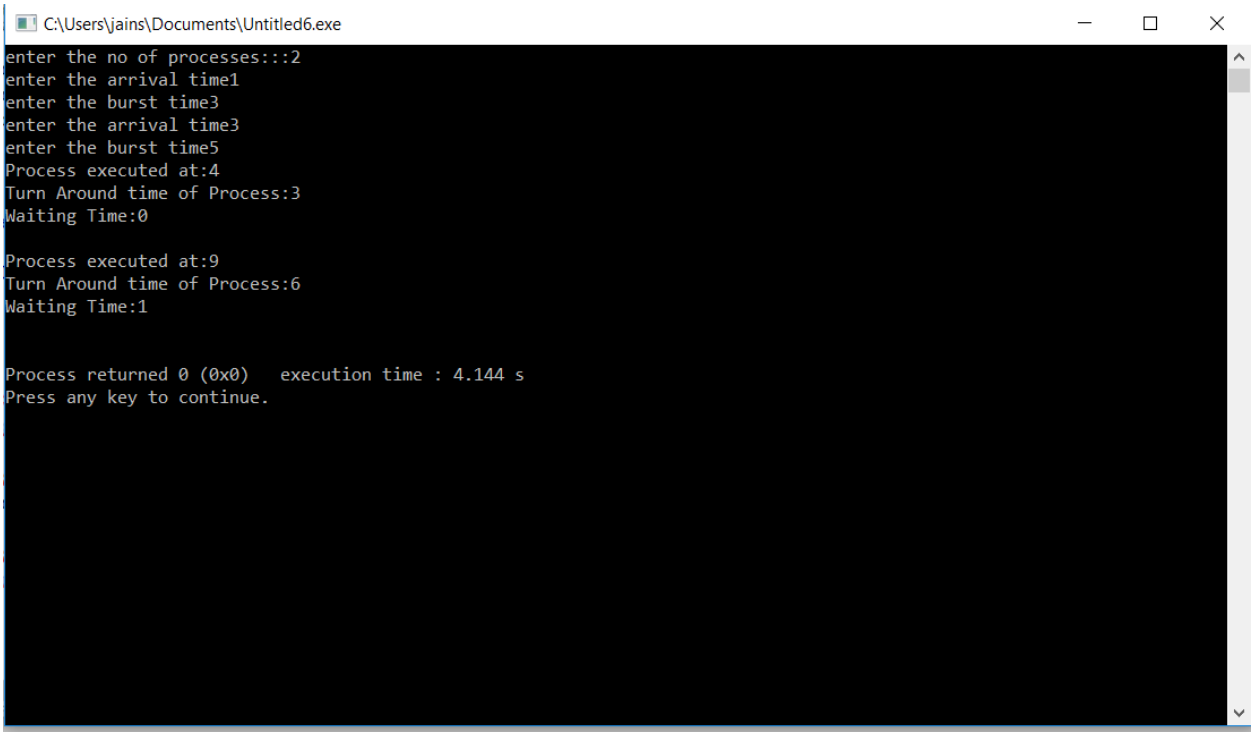
Turn around time= Compilation time – arrival time

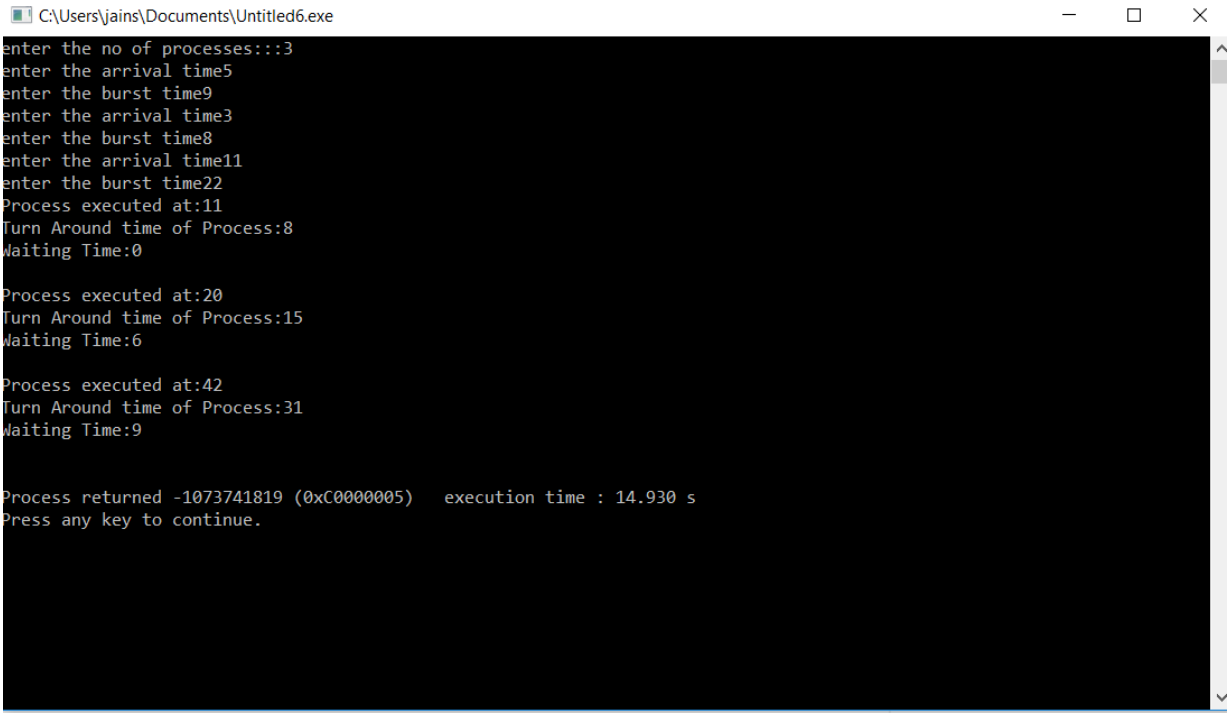
Waiting time = Turnaround time – Burst time

Average Waiting time = total waiting time / no of processes

Average Turn around time = total burst time/ no of processes







These are the snippets of the program in which no processes 3,4,2.

6. If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.

Ans I implement Selection Sort algorithm in additional …it ios required to sort the linked according to their corsponding arrival time …which make us easy to analuyze which process arrive at first..and using linked list concept ..