Name :- Pritika Chaudhary
Course :- B.Sc (it) 2
Section :- B
University Roll No :- 2023078
Student Id :- 20052056
Exam :- Mid term Practical Exam (PBI-202)
Date :- 22 June, 2021

Q1):

```
#include <stdio.h>
int heap[100], ind[100], pos[100], size = 0;
int temp[100]; temp1[100];
int arr_time[100], cook_time[100], n;
void merge(int low, j = mid+1, k=0;
while(i<=mid && j<=high)
{
if(arr_time[i] <= arr_time[j])
{
temp[k] = arr_time[i];
temp1[k] = cook_time[i];
i++;
k++;
```

```
if(i<=mid)
{
int i;
for(i=l; i<=mid; i++)
{
temp[k]=arr_time[i];
temp1[k]=Cook_time[i];
k++;
}
else if(j<=high)
{
int i;
for(i=j; i<=high; j++)
{
temp[k]=arr_time[i];
temp1[k]=Cook_time[i];
k++;
}
}
void divide(int low, int high)
{
if(low<high)
{
int mid=(low+high)/2;
divide(low, mid);
divide(mid+1, high);
merge(low, mid, high);
```

```
void insert(int node, int val)
{
int s;
if(has[node]==0)
{
heap[++size]=val;
ind[size]=node;
has[node]=size;
s=size;
}
else
{
heap[has[node]]=val;
s=has[node];
}
while(s!=1)
{
if(heap[s/2]>heap[s])
{
int t=heap[s/2];
heap[s/2]=heap[s];
heap[s]=t;
t=ind[s/2];
ind[s/2]=ind[s];
ind[s]=t;
has[ind[s/2]]=s/2;
has[ind[s]]=s;
}
else
break;
```

```
    s=s/2;
}
}
int extract_min()
{
int n = ind[1];
int s=1;
has[n] = -1;
ind[1] = ind[size];
has[ind[size]] = 1;
heap[1] = heap[size-1];
while(1)
{
int t;
if heap[s*2]<heap[s] && s*2<=size || heap[s*2+1]<heap
                                        [s] && s*2+1)
{
if(heap[s*2] < heap[s*2+1]
t = s*2;
else
t = s*2+1;
int t = heap[t];
heap[t] = heap[s];
heap[s] = t;
t = index[t];
index[t] = index[s];
index[s] = t;
has[ind[t]] = t;
has[ind[s]] = s;
}
else
break;
```

```
=T;
return n;
}
void init(int n)
{
int i;
for(i=1; i<=n; i++)
{
hash[i]=0;
vid[i]=0;
heap[i]=[100];
}
size=n;
}
int main()
{
int A_T, C_T, i=1;
long wait_time=0, time=0;
scanf("%d", &n);
for(i=0; i<n; i++)
scanf("%d%d", &arr_time[i], &Cook_time[i]);
divide(0, num-1);
for(i=n; i>=1; i--)
{
arr_time[i]=arr_time[i-1];
Cook_time[i]=Cook_time[i-1];
}
insert(1, Cook_time[1]);
i=2;
while(i<=n && arr_time[i]==arr_time[1])
{
```

```
insert(i, Cook_time[i]);
i++;
}
while(size!=0)
{
int i = Extract_min();
if (time>arr_time[i])
{
wait_time+= time - arr_time[i]+ Cook_time[i];
time+=Cook_time[i];
}
else
{
time=arr_time[i]+Cook_time[i];
wait_time+= Cook_time[i];
}
printf("%d %d %d \n", i, time, wait_time);
i=i
while(i<= heap[i] <=time)
{
return 0;
}
}
```

Q2) Each process p_1 p_2 p_3 p_4 with arrival time and cpu time respectively (0, 10) (0, 2) (0, 4). Write a C program to implement SJF algorithm and find out turn around time and average waiting time.

Coding

```
#include <stdio.h>
int main()
{
int bt[20], p[20], wt[20], tat[20], i, j, n, total=0;
int pos, temp;
float avg_wt, avg_tat;
printf("Enter number of process: ");
scanf("%d", &n);

printf("\nEnter Burst Time: \n");
for(i=0; i<n; i++)

printf("p%d: ", i+1);
scanf("%d", &bt[i]);
p[i] = i+1;

for(i=0; i<n; i++)
```

```
{
has = i;
for(j = i+1; j < n; j++)
{
if(bt[j] < bt[has])
  has = j;
}
temp = bt[i];
bt[i] = bt[has];
bt[has] = temp;
temp = p[i];
p[i] = p[has];
p[has] = temp;
}
wt[0] = 0;

for(i=1; i<n; i++)
{
wt[i] = 0;
for(j=0; j<i; j++)
  wt[i] += bt[j];
total += wt[i];
}
```

```
avg_wt = (float)tot/n;
tot = 0;
printf("\n Burst time \t waiting time \t turnaround time");
for(i=0; i<n; i++)
{
tot[i] = bt[i] + wt[i];
tot += tot[i];
printf("\n %d %d %d %d", p[i], bt[i], wt[i], tot[i]);
}
avg_tat = (float)tot/n;
printf("\n Average Waiting Time = %f", avg_wt);
printf("\n Average Turnaround Time = %f", avg_tat);
}
```

Output

Enter the no. of process: 4

Enter pname, arr time & execution time = p1 0 10

Enter pname, arr time & execution time = p2 0 2

Enter pname, arr time & execution time = p3 0 1

Enter pname, arr time & execution time = p4 0 4

h name	arrival time	exec time	waiting time	tat time
			0	1
h3	0	1	1	3
h2	0	2	3	7
h4	0	4	7	17
h1	0	10		

Average waiting time is: 2.7500
Average turnaround time is: 7.000