DATE

PAGE NO.

Name - Amar Singh Bhandari
Student id - 2005 1023
University R.N - 2023027

~~Ans~~ 1)
```
# include <stdio.h>
unsigned int
Heap[100001], Index[100001], Position[10001], size=0
usinged int
Temp[100001], Temp1[100001];
unsigned int
~~Temp[100001]; Temp1[100001];~~
ArrTime[100001], cookTime[100001], Num;
void merge(int low; int mid, int High)
{
 int i=low; j= mid+1, k=0;
while(i<= mid && j<= High)
{
if(ArrTime[i]<=ArrTime[j])
{
Temp[k] = ArrTime[i];
Temp1[K] = cookTime[i];
i++;
k++;
}
else
{
 Temp[k] = ArrTime[j];
```

SONA

TEACHER'S SIGNATURE

DATE ....

PAGE NO......................

```
j++;
k++;
}
}
if(i>=mid)
{
 int l;
for (l=i; l<=mid; l++)
{Temp[k] = cook Time[l]; k++;}
}
 else if(j<=High)
{
 int l;
for (l=j, l<= High; l++)
{Temp[k]= Arr Time[l];
Temp1[k] = cook Time[l]; k++;}
}
 k = 0;
 for(i=Low; i<=High; i++)
{
 Arr_Time[i] = Temp[k];
cook Time[i]=Temp1[k];
k++;
}
  }
void divide(int low, int High)
{
```

SONA

TEACHER'S SIGNATURE..........................

DATE ....

PAGE NO....................

```
if(Low<High)
{
 int mid=(Low+High)/2;
divide (Low, mid);
divide (mid+1, High);
merge(Low ,mid, High);
}
  }

void Insert(int Node, unsigned int value)
{
int s;
if(position[Node]==0)
{
Heap[++size]=value;
Index[size]= Node;
position[Node] = size;
S= size;
}
else
{
Heap[position[Node]];
}
 while(s!=1)
{
 if(Heap[s/2]>Heap[s])
{
```

SONA

TEACHER'S SIGNATURE........................

DATE ....

PAGE NO....................

```
int t = Heap [S/2];
Heap[S/2]= Heap[S];
Heap [S] = t;
t = Index[S/2];
Index[S/2]= Index[S];
Index [S]= t
Position[Index[S/2]] = S/2;
position[Index[S]] = S;
}
else
break
S = S/2;
}
}
int Extract min()
{
 int N = Index[1];
int S = 1;
// printf ("%d\n", Heap[1]);
position[N] = -1;
Index[1] = Index [size];
position[N] = -1
Heap[1] = Heap[size--];
while(1)
{
 int T;
if(Heap[S*2] < Heap[S] & S*2 <= size||
```

SONA

TEACHER'S SIGNATURE..............

DATE .... PAGE NO.....................

```
{
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;
position[Index[T]]=T;
position[Index[S]]=S;
}
else
break;
S=T;
}
return N;
}
 voindInt (int N)
{
 int i;
For(i=1;<N;i++)
{
position[i]=0;
Index[i]=0;
```

SONA

TEACHER'S SIGNATURE...

DATE .... PAGE NO.....................

```
Heap[i] = 10000000l;
}
size=N;
}
int maid()
{
int A.T.C-Ti =1;
int l= Extract min();
if(Time) Arr Time[l])
{
wait Time+= Time-Arr Time[l]+ cook Time[l];
Time += cook Time[l]
// printf("%d %d %d \n",l,Time, wait Time);
}
else
{
Time =Arr- Time[l]+ cook Time[l];
wait Time+= cook Time[l];
}
// printf("%d%d %lld %lld \n",l,Time, wait Time);
l=i;
}
  Insert(i,Cook Time[i]);
l++;
}
```

SONA TEACHER'S SIGNATURE.....................

DATE ....

PAGE NO....................

```
while (i<=num&&ArrTime[i]==ArrTime[i])
{
insert(i,cookTime[i]);
i++;
}
}
    }
waitTime = wait_Time/num; ++i
 printf("%lld", wait_Time);
// system("puse");
return 0;

 }
```

SONA

TEACHER'S SIGNATURE..........................