NAME - Brijendra Uniyal
Roll No - 2005070
Course - BSc IT
Semester - 2nd.

Subject - Operating System. ①

Q-① - #include<stdio.h>
unsigned int
Heap[100001], Index[100001], Position[10000], Size=0;
unsigned int Temp[100001], Temp1[100001];
Unsigned int
Arr_Time[100001], Cook_Time[100001], Num;
Void Merge(int Low, int Mid, int High)
{
int i=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++;
}
else.
{Temp[K]=Arr_Time[j];
Temp1[K] = Cook_Time[j];
j++;
K++;
}

Brijendra
22/06/2021

```
}
if (i<=Mid)
{int l;
 for (l=i; l<=Mid; l++)
 {Temp[k] = Arr-Time[l];
  Temp1[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)
{
```

Priyanka
22/06/2021

```
If (Low < High).
{
Find 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]] = Value;
    S = Position (Node);
   }
   while ( S != 1).
{ If (Heap [S/2] > Heap [S])
  { int t = Heap [S/2];
    Heap [S/2] = Heap (S);
    Heap [S] = t;
```

Priyanka
22/06/2021

```
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[Index[Size]]=1;
Heap[1]=Heap[Size--];
while(1)
{int t;
If(Heap[s*2]<Heap[s]&&s*2<=Size||
Heap[s*2+1]<Heap[s]&&s*2+1<=Size)
{
If(Heap[s*2]<Heap[s*2+1]).
t=s*2;
```

Priyanshu
22/06/2021

```
else.
r = s*2+1;
int t=Heap[r];
Heap[r] = Heap[s];
Heap[s]=t;

t= index[r];
Index[r] =Index[s];
Index[s]=t;

Position[Index[r]]=r;
Position[Index[s]]=s;
}
else.
break;
s=r;
}
return N;
}
Void int(int N)
{ int i:
for(i=1: i<= N+1)
{ Position[i]=0;
  index[i] =0;
  Heap[i] = 100000001;
} size = N;
}
int main()
```

Ritzanshy
22/01/2021

```
{ int A_T, (_T, i=1;
  long long Wait _Time = 0, Time =0;
  scanf ("%d", & Num);
  // init (N);
  for (i=0; < Num; i++)
  scanf ("%d%d", & Arr_Time [i], & Work_Time[i]);
  divide (0, Num-1);
  for (i= Num; i>= i; i--)
  { Arr_Time(i] = Arr_Time[i-1]);
    }
    Insert ( 1, Work_Time(1)];
    i=1;
    while (i2=Num && Arr_Time[i] == Arr_Time[1]
    {
    Insert (i Work_Time[1]);
     i++;
    }
while (size!=0)
{ int 1= Extract_Min ();
if (Time > Arr_Time [1]).
{ Wait Time t = Time Arr_Time [1] + Work_Time [1];
  Time t = Work_Time [1];
// printf ("%d %d %d\n", 1, Time, Wait_Time);
1 = i;
```

Priyanshu
22/08/2021

(2)-

```
while(i<=Num && Arr_Time[i]<=Time).
{
insert(i, look_Time[i]);
i++;
}
if(i==i && i<=Num) // No job is before curr_time.
{
Insert(i, look_Time[i]);
i++;
}
If (i==0 && i<=Num) // No job id before curr_time.
{ insert[i look_Time[i]];
i++;
while(i<=Num && Arr_Time[i] == Arr_Time[i]).
{
insert(i, look_Time[i]);
i++;
}
}
}
Wait_Time = Wait_Time / Num;
Printf("%lld", Wait_Time);
// System("Pause");
return 0;
```

3.

Priyanshu
22/06/2021

```
        {
        Insert(i,Cook_Time[i]);
        i++;
        }
        if(I==i&&i<=Num)//No job is before curr_time
        {
          Insert(i,Cook_Time[i]);

          i++;
        while(i<=Num&&Arr_Time[i]==Arr_Time[I])
              {
               Insert(i,Cook_Time[i]);
               i++;
              }
              }
    }
    Wait_Time=Wait_Time/Num;
    printf("%lld",Wait_Time);
   // system("pause");
```