Name - Vikas Kumar

University Roll no - 2023113

Q1. Ans. Code

```
#include<stdio.h>
unsigned int
Heap [100001], Index [100001], Position [100001], Size=0;
unsigned int
Temp[100001], Temp 1 [100001];
unsigned int
Arr_Time [100001], Look_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];
    Temp 1 [k] = Look_Time [i];
    i++;
    k++;
   }
   else
   {
    Temp [k] = Arr_Time [j];
```

[illegible]

```
Temp1[k]=Cook_Time[j];
 j++;
 k++;
 }
}
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)
{
 if(low < High)
  {
    int Mid=(Low+High)/2;
```

V.K

```
    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;

      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;
      else
      T=s*2+1;
      int t= Heap[T];
      Heap[T] = Heap[S];
      Heap[S]= t;
```

V.K

```
t= Index [T];
 Index [T] = Index [S];
 Index [S]= t;

 Position [Index [T]] = T;
 Position [Index [S]] = S;
}
else
break;
S=T;
}
return N;
}
void init (int N)
{
 int i;
for (i=1; i <=N; i++)
{
 Position [i] =0;
  Index [i]=0;
  Heap [i]= 1000000001;
  }
   Size N;
}
int main()
{
 int A_T, C_T, i=1;
 long long Wait_Time = 0, Time =0;
 scanf("%d", &NUM);
 //int (N);
for (i=0; i<NUM; i++)
scanf("%U%U", & Arr_Time [i], & Cook_Time [i]);
```

V.K

```
divide(0,Num-1);
for(i=Num; i>=1; i--)
{
     Arr_Time[i]=Arr_Time[i-1];
 Cook_Time[i]=Cook_Time[i-1];
//Printf("%v %v\n", Arr_Time[i], Cook_Time[i]);
}
Insert(1, Cook_Time[1]);

i=2;

While(i<=Num&& Arr_Time[i] == Arr_Time[1])
   {
    insert(i, Cook_Time[i]);
    i++;
   }
  while(size!=0)
  {
    int l=Extract_Min();
    if(Time>Arr_Time[l])
    {
     wait_Time+=Time-Arr_Time[l]+_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;
```

V.K

```
while (i<=Num&&Arr_Time [i] <=Time)
  {
   insert(i, Cook_Time[i]);
   i++;
  }
  if (1==i&&i<=Num) //No job is before curr_time
   {
    Insert (i,Cook_Time[i]);

    i++;

 while (i<=Num&& Arr_Time [i]== Arr_Time [1])

     {
       Insert (i,Cook_Time [i]);

      i++;

     }
    }
 }
  wait_time = Wait_Time /Num;
  printf("%.lld", Wait_Time);
 //system("pause");
  return0;
}
```

V.K