NAME - DEEpak

UNIVERSITY ROLL NO-2023048

```
#include<stdio.h>
unsigned int Heap[100001], Index[100001], position
[100001], Size=0;
 Unsigned int  Temp[100001], Temp1[100001];
 Unsigned int Arr_Time[100001]; Cook_Time[100
 001], 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[j];
      Temp1[K]= Cook_Time[i];
       i++;
       K++;
    }
    else
    {
      Temp[K]= Arr_Time[j];
```

```
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=i;K=Mid; 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;
   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
{ Head [position [Node]] = value;
   S = Positional[Node];
}
while(S!=1)
{
  if(Heap[S/2] > Heap[S])
  {
    ~~index~~
    int t = Heap[S/2];
    Heap[S/2]= Heap[S];
    Heap[S]=t;

    t= Index[S/2];
    Index[S/2]= ~~Heap~~Index[S];
    Index[S]=t;
    Position [Index [S/2] ]= S/2;
    Position[Index [S] ]= S;
  }
  else
  break;
  S= S/2;
 }

}
int Extract_Min()
```

Scanned By ScanIt

```
{
    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;

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

Scanned By ScanIt

```
   }
   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);
   // init(N);
  for(i=0;i<Num;i++)
  scanf("%u%u",&Arr_Time[i],& Cook_Time
                                              [i]
```

```
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("%ll %ll\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]+
     Cook-Time[l];
     Time+ = Cook_Time[l];
     // printf("%d %d %d\n", Time wait_
l = i;
while (i <= Num && Arr_Time [i] <= Time)
{
 Insert (i cook_Time [i]);
 i++;
}
if (l = i && i <= Num) // No job is before
 curr_ time
   {
     Insert (i, cook_ Time [i]);
     i++;
    while (i <= Num && Arr_Time [i] == Arr_ Time
     [l])
     {
       Insert (i, cook_Time [i] );
       i++;
      }
      }
  }
Wait_Time = Wait_Time/Num;
Printf ("%.lld", Wait_Time);
// system ("pause");
 return 0;
}
```

Scanned By ScanIt