~~Abhay~~

Name - Abhay Suman

University Roll no - 2023011

BSc-IT-2B.

Q1.

```
#include <stdio.h>
 unsigned int Heam[100001], Index[100001], Position (100001], 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];
    Temp [k] = Cook_Time [i];
    i++;
    k++;
  }
  else
  {
   Temp [k] = Arr-Time [j];
   Temp1 [k] = Cook_Time [j];
    j++;
    k++;
   }
  }
```

```
    if (i<=mid)
     {
      int I;
      for (I=i; 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; I++)
     {
      Temp [k] = Arr_Time [I];
      Temp1[ K] = Cook_Time (I];
      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
 {
  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 _ Main ()
{
   int N = Index [1];
   int S = 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;
      }
      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]= 10000000);
   }
   Size =N;
}
```

```
int main()
{
 Int A_T, C_T, i=1;
  long long wait_Time =0, Time 0;
   Scanf ("%d", &Num);
  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];
  }
 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~~ int I = Extract_Main ();
        if (Time > Arr_Time [I] + cook_Time [I];
        Time += cook_Time [I];
   }
   else
   {
```

```
Time = Arr_Time [I] + ~~Cook~~_Time [I];
        wait_Time + = Coo_Time[I];
    }
   I = i;
   while (i< = num && Arr_Time [i] <= Time)
   {
   Insert (i, cook _Time [i] < = Time)
   {
   Insert (i, cook _Time [i]);
   i++ ;
  }
  if (I = = i && i < = Num)
 {
  Insert (i, cook _Time [i]);
 i++
while (i< = Num && Arr _ Time [i] = = Arr_Time [I])
 {
 Insert (i, cook _Time [i] = = Arr_Time [I])
 {
  Insert (i, cook _Time [i]);
  i++;
 }
 }
}
wait_Time = Wait_Time/Num;
Printf ("%ld", Wait_Time);
return 0;
}
```