Name :- Shubham Singh Soragi

Course:- Bsc(IT)

Student ID:- 20051086

Rollno:- 2093008

Campus:- Haldwani

Q1.

```
#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[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++;

   }

  }
  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_Min()

{

    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]=1000000001;
  }
  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 I=Extract_Min();
          if(Time>Arr_Time[I])
          {
           Wait_Time+=Time-Arr_Time[I]+Cook_Time[I];
           Time+=Cook_Time[I];
          }
          else
          {
              Time=Arr_Time[I]+Cook_Time[I];
              Wait_Time+=Cook_Time[I];
          }
          I=i;
          while(i<=Num&&Arr_Time[i]<=Time)
          {
          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);
    return 0;
}
```

OUTPUT-