Name: Mohit Rana
University Roll.NO: 2023067
Section: A
Course: BSc.It

Ques:t
Ans

```
#include<stdio.h>
unsigned int Heap[100001], Index[100001], Position[100001],Size
unsigned int Temp[100001], Temp1[10001];
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++
```

Mohit

```
}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;
    }
```

Mohit

```
{
 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;
```

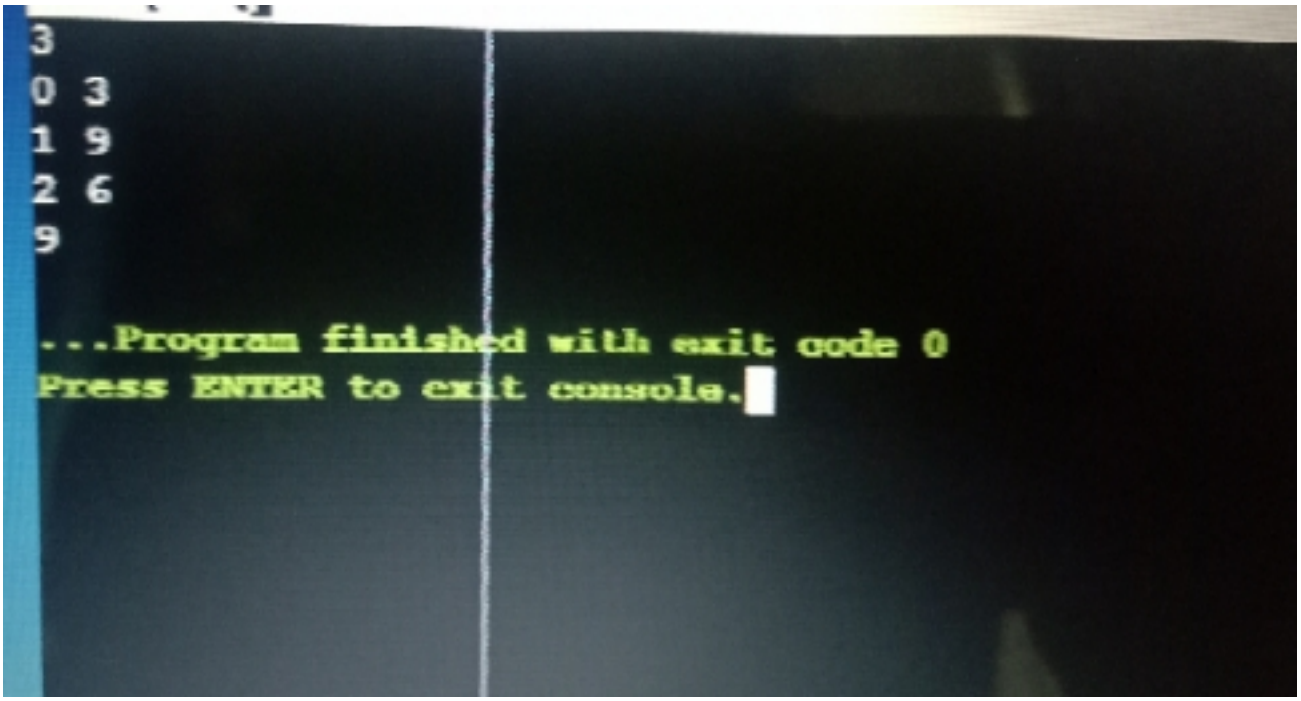
Mohit

```
}
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++)
 Scan f("%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_Tisme [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];
   }
```

Mohit

```
I = i;
while (i <= Num && Arr_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]);
  i++;
 }
 }
}
 wait Time = Wait_Time / Num;
 printf ("%.1lf", Wait_Time);
 return 0;
}
```

Mohit


3
0 3
1 9
2 6
9
...Program finished with exit code 0
Press ENTER to exit console.