Name - Akkimaan
Roll no - 2023023
Student ID = 2005107

Q-1
=

```
#include <stdio.h>
unsigned int
Heap [100001], Index [100001], Position [10000]
, size = 0;
unsigned int
Temp [100001], Temp1 [10000];
unsigned int
Arr_Time [100001], Cook_Time [10000]
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++;
}
```

classmate
Date
Page

```
else
{
  Temp[k] = Am_Time[j];
  Temp1[k] = Cook_Time[j];
  j++;
  k++;
 }
}
 if (i<=mid)
{
  int l;
  for (l=i; l<=mid; l++)
   {Temp[k] = Am_Time[l];
Temp1[k] = Cook_Time[l]; k++;}
}
 else if (j<= High)
{
 int l;
 for (l=j; l<=High; l++)
  {Temp[k] = Am_Time[l];
Temp1[k] = Cook_Time[l]; k++;}
}
  k = 0;
for (i=low; i<= High; i++)
{
  Am_Time[i] = Temp[k];
  Cook_Time[i] = Temp2[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;
```

```
{
 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];
    Ind>c[s/2] = index [s];
   Index [s]= t;

    Position [Index [s/2]]= s/2;
     Position [Index[ 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;

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

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

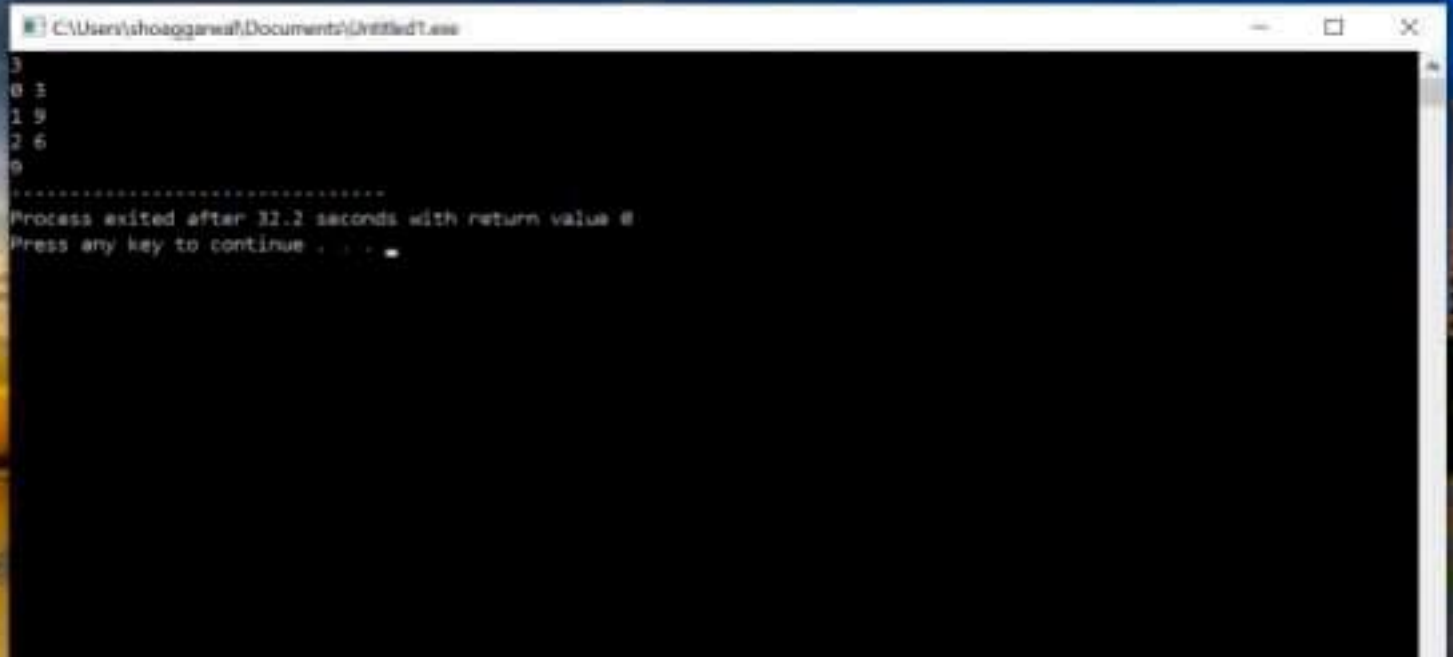
Date
Page

```
}
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, Temp=0;
scanf("%d", &Num);
// init(N);
for(i=Num; i>=1; i--)
{
```

```
Asn_Time[i]=Asn_Time[i-1];
Cook_Time[i]=Cook_Time[i-1];
//Printf("%d %d
%d %d\n", Asn_Time[i], Cook_Time[i]);
}
Insert(1, Cook_Time(1));
i=2;

while(i<=Num && Asn_Time[i]=Asn_
Time[1])
    {
      Insert(i, Cook_Time[i]);
      i++
    }
while(size!=0)
 {
   int t=Extract_min();
   if(Time>Asn_Time[1])
{
  Wait_Time t=Time-Asn_Time[1]+
Cook_Time[1];
     Time t=Cook_Time[1];
   //Printf("%d %d %d %d\n", 1, Time,
   Wait_Time);
}
else
 {
```

```
Time = Arr_Time[i]
+ Cook_Time[i];
    wait_Time t = Cook_Time[i];
}
// Printf("%d %d %d\n", Time
wait_Time);
    i = i;
while(i<=Num && Arr_Time[i]<=Time)
{
    Insert(i, Cook_Time[i]);
    i++;
}
if(i== i && i<=Num) // No job is
before cum_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);
// system("Pause");
return 0;
}
```


C:\Users\shoaggarwal\Documents\Untitled1.exe
3
0 3
1 9
2 6
0

Process exited after 32.2 seconds with return value 0
Press any key to continue . . .