```
# include <Stdio.h>
unsigned int heap [100001] , Index [100001] , Position [100001]
        size = 0;
unsigned int temp [100001] , Temp [10000 1];
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;

//  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;
```

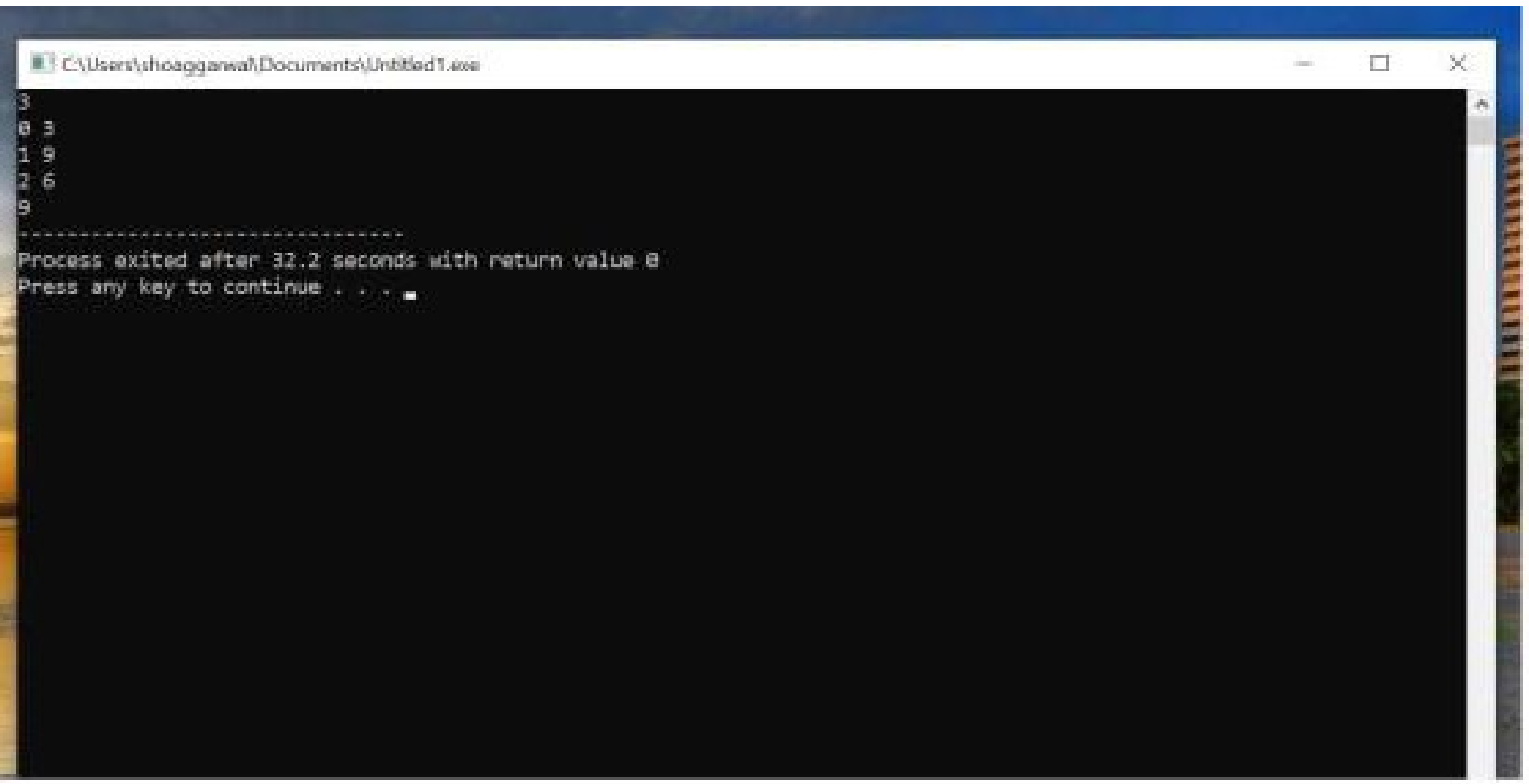
```
        position[Index[T]] = T;
        position[Index[S]] = S;
    }
    else
    break;
    S = T;
}
    return N;
}
void Init (int N)
{
    int i;
    for (i=1; i<N; i++)
    {
        position[i] = 0;
        Index[i] = 0;
        Heap[i] = 1000000001;
    }
    int main  Size = N;
}
    int 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 ("%d%d", 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("%u %u\n", Arr_Time[i], Cook_Time[i]);
    }
    Insert(1, Cook_Time[1]);

    i=2;
        while (i<=Number && Arr_Time[i]==Arr_Time[i])
    {
        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];
// printf("%d %d %d\n", I, Time, Wait_Time);
    }
    else
    {
        Time = Arr_Time[I] + Cook_Time[I];
        Wait_Time += Cook_Time[I];
    }
    // printf("%d %lld %lld\n", I, Time, Wait_Time);
        I = i;
        while (i<=Number && 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 ("%Ld", Wait_Time);
    //System ("pause");
    return 0;
  }
```


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

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