Question 1:

```
#include <stdio.h>
unsigned int heap [10001], Index [10001], Position [10001], Size = 0;
unsigned int Temp [10001], Temp1 [10001];
unsigned int Arr_Time [10001], Cook_Time [10001], 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];
    Temp 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<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 (S1 = 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
  }
Extract - Min ( )
int N= Index [1];
 int s=1;
// print f(" %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 )
{
 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);
  //int (N);
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];
  // printf ("%u, %u \n", Arr_Time [i], Cook_Time [i]);
  }
  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 t = (cook -Time [I];
// print ("%.d %.d %.d \n", I, Time, Wait - Time);
 }
 else.
 {
 Time = Arr - Time [I]+(cook -Time (I];
 Wait - Time + = (cook - Time [I];
 }
// print f("%.d %. 11d %.11d \n", I, Time, Wait - Time);
I = i;
while (i<= Nom && Arr - Time (i] <= Time)
```

Prem