```
#include<bits/stdc++.h>
using namespace std;
vector<pair<int,int>>v;
queue<int> q1,q2;
int input(int number)
{
    cout<<"Input Entry time and Service time
: \nE-T S-T\n";
    for(int i=0; i<number; i++)
    {
        int Enter_time,Srvc_time;
        cin>>Enter_time>>Srvc_time;

v.push_back(make_pair(Enter_time,Srvc_time));
    }
    sort(v.begin(),v.end());
}
int  calculation(int number)
{
    if(number>=2)
    {

    int srvc1=0,srvc2=0;
    q1.push(1);
    q2.push(2);
    srvc1+=v[0].first+v[0].second-1;
    srvc2+=v[1].first+v[1].second-1;
    for(int i=2; i<number; i++)
    {
        if(srvc1<=srvc2)
        {
            q1.push(i+1);
            srvc1+=v[i].second;
        }
        else
        {
            q2.push(i+1);
            srvc2+=v[i].second;
        }
    }
    }
    else
    {
        q1.push(1);
    }
}
```

```
void result()
{
    cout<<"First Queue : ";
    while(!q1.empty())
    {
        cout<<q1.front()<<"
";
        q1.pop();
    }
    cout<<endl;
    cout<<"Second Queue : ";
    while(!q2.empty())
    {
        cout<<q2.front()<<"
";
        q2.pop();
    }
    cout<<endl;
}
int main()
{
    cout<<"Enter the number
of people getting services :
";
    int number;
    cin>>number;
    input(number);
    calculation(number);
    result();

    return 0;
}
```

```
void InsertionSort(int ara[],int n)
{
    for(int i=1; i<n; i++)
    {
        int j=i-1;
        int temp=ara[i];
        while(j>=0 && ara[j]>temp)
        {
            ara[j+1]=ara[j];
            j--;
        }
        ara[j+1]=temp;
    }
}

void BubbleSort(int ara[],int n)
{
    for(int i=0; i<n-1; i++)
    {
        for(int j=0; j<n-1-i; j++)
        {
            if(ara[j]>ara[j+1])
            {
               swap(ara[j],ara[j+1]);
            }
        }
    }
}

void SelectionSort(int ara[],int n)
{
    for(int i=0; i<n-1; i++)
    {
        int temp;
        for(int j=i+1; j<n; j++)
        {
            if(ara[i]>ara[j])
            {
                temp=j;
            }
        }
        if(temp!=i)
        {
            swap(ara[temp],ara[i]);
        }
    }
}
```

```
int Partition(int ara[],int
left,int right)
{
    int i,j,pivot;
    i=left;
    j=right;
    pivot=ara[left];
    while(i<j)
    {
        while(ara[i]<=pivot)
        {
            i++;
        }
        while(ara[j]>pivot)
        {
            j--;
        }
        if(i<j)
        {
            swap(ara[i],ara[j]);
        }
    }
    swap(ara[left],ara[j]);
    return j;
}

void QuickSort(int ara[],int
left,int right)
{
    if(left>=right)
    {
        return ;
    }
    int
p=Partition(ara,left,right);
    QuickSort(ara,left,p-1);
    QuickSort(ara,p+1,right);
}
```

```
int Merge(int ara[],int left,int
mid,int right,int n)
{
    int temp[n];
    int i=left;
    int   j=mid+1;
    int k=left;
    while(i<=mid && j<=right)
    {
        if(ara[i]<=ara[j])
        {
            temp[k]=ara[i];
            i++;
            k++;
        }
        else
        {
            temp[k]=ara[j];
            j++;
            k++;
        }
    }
    if(i>mid)
    {
        while(j<=right)
        {
            temp[k]=ara[j];
            j++;
            k++;
        }
    }
    else
    {
        while(i<=mid)
        {
            temp[k]=ara[i];
            i++;
            k++;
        }
        for(int k=left; k<=right;
k++)
        {
            ara[k]=temp[k];
        }
    }

}
```

```
  void MergeSort(int ara[],int
left,int right,int n)
{
  if(left<right)
    {
        int mid=(left+right)/2;
        MergeSort(ara,left,mid,n);

MergeSort(ara,mid+1,right,n);

Merge(ara,left,mid,right,n);
    }
}
```