Problem statement:

> Dominos Pizza has hungry customers waiting in the queue. Each unique order is placed by a customer at time x[i], and the order takes y[i] units of time to complete.
> You have information on all n orders, Find the order in which all customers will receive their pizza and return it. If two or more orders are completed at the same time then sort them by non-decreasing order number.

Example:

Input :

```
arr[ ] = {{4,1}, {6,2}, {7,6},{8,1}, {1,3}}
```

Output :

```
5 1 2 4 3
```

Explanation:
Here an array can be transform to
{5, 8, 13, 9, 4}. Here 5th index order
received first then 1st order, 2nd order ...
return {5, 1, 2, 4, 3}

Approach:

⟶ So here we will store the sum of `x[i]` and `y[i]` in one array and also we will store their 1 based index in that array so basically we need vector of pairs.

```
vector<pair<int,int>>temp(n);
for(int i=0;i<n;i++){
```

```
        temp[i] = make_pair(arr[i][0] + arr[i][1],i+1);
    }
```

After making the vector we will sort it in increasing order and **if 2 pairs have same first element then we will sort them on the basis of second element.**

```
bool cmp(pair<int,int>&a,pair<int,int>&b){
    if(a.first == b.first){
        return a.second<b.second;
    }
    return a.first<b.first;
}
vector<int> permute(vector<vector<int>> &arr, int n)
{

    vector<pair<int,int>>temp(n);
    for(int i=0;i<n;i++){
        temp[i] = make_pair(arr[i][0] + arr[i][1],i+1);
    }
    sort(temp.begin(),temp.end(),cmp);
    vector<int>ans;
    for(int i=0;i<n;i++){
        ans.push_back(temp[i].second);
    }
    return ans;
}
```

At last we will return the `ans` vector which will store the 1 based indexes of all orders.