FED : DEPARTMENT OF BASIC ENGINEERING SCIENCE – I SUBJECT CODE: 21SC1202
DATA STRUCTURES WORKBOOK

WEEK - 1

Lab Session:

Date of the Session: ___/___/___ **Time of the Session:** ___

Prerequisite:

- Divide and Conquer Algorithm
- Sorting
- swapping items in place
- Partitioning of the array.

Pre-lab Task:

. In Indian battalion 35 – NCC Brigadier Vishal wants to sort his troop's corps in ascending ord
he corps height. So, help Brg.Vishal to arrange corps accordingly. (Imagine corps height as ele
Analyze the time complexity**(O(n log n))**

Hint: Insertion sort helps to sort array in best time complexity.

Program :-

```
#include<stdio.h>
void insrt (int a[], int n)
{
    int i,j,k,temp;
    for(i=1;i<n;i++)
    {
      for(j=i;j>0;j--)
      {
        if(a[j]<a[j-1])
        {
          temp=a[j];
          a[j]=a[j-1];
          a[j-1]=temp;
        }
      }

      for(k=0;k<n;k++)
          Printf("%d ",a[k]);
}
    Printf("\n");
}
```

```
int main()
{
    int n, i, m;
    scanf("%d",&n);
    int a[n];
    for(i=0; i<n; i++)
     scanf("%d",&a[i]);
  for( m=0; m<n ;m++)
          Printf("%d ", a[m]);
      Printf("\n");
   insrt(a,n);
}
```

2. Given a list of N array element apply shell sort.. The first line contains an integer, N, the number of elem in Array. The second line contains N space-separated integers. Print the array as a row of space-sepa integers in each iteration.

Program :-

```
#include<stdio.h>
Void shell(int a[], int n)
{
 int i,j,temp,g,P;
 for(g=n/2; g>=1; g=g/2)
 {
  for(i=g; i<n; i++)
  {
   for(j=i; j>=g; j=j-g)
   {
    if(a[j]<a[j-g])
    {
     temp=a[j];
     a[j]=a[j-g];
     a[j-g]=temp;
    }
   }
   for(P=0; P<n; P++)
    Printf("%d", a[P]);
    Printf("\n");
  }
}
int main()
{
 int n;
 scanf("%d",&n);
 int a[n],i;
 for(i=0; i<n; i++)
 scanf("%d",&a[i]);
 shell(a,n);
}
```

In-lab Task:

1) Implement Insertion Sort

Given a list of N array elements apply insertion sort on array.

Input Format

The first line contains the integer N, the size of the array .

The next line contains N space-separated integers .

Constraints

- 1<= N <= 1000
- -1000<= a[i]<= 1000

Output Format: Print the array as a row of space-separated integers each iteration .

https://www.hackerrank.com/contests/17cs1102/challenges/3-a-implement-insertion-sort

Program :-

```
#include<stdio.h>
void insrt(int a[], int n)
{
   int i,j,k,temp;
   for(i=1;i<n;i++)
   {
      for(j=i;j>0;j--)
      {
         if(a[j]<a[j-1])
         {
            temp=a[j];
            a[j]=a[j-1];
            a[j-1]=temp;
         }
      }
      for(k=0;k<n;k++)
         Printf("%d ",a[k]);
      Printf("\n");
   }
}
int main()
{
   int n,l,m;
   Scanf("%d",&n);
   int a[n];
   for(l=0;l<n;l++)
```

```
    scanf("%d", &a[i]);
for(m=0; m<n; m++)
        printf("%d ", a[m]);
    printf("\n");
    insrt(a,n);
}
```

2. Quick Sort

Given a list of N array elements apply Quick Sort.

Input Format

The first line contains an integer, N, the number of elements in Array.

The second line contains N space-separated integers.

Constraints

1 <= N <= 1000

-1000 <= array elements <= 1000

Output Format

Print the array as a row of space-separated integers each iteration

https://www.hackerrank.com/contests/17cs1102/challenges/4a-quick-sort

Program :-

```
#include<stdio.h>
Void quick_Sort(int arr[100],int,int);
int Partition(int arr[100],int,int);
int main()
{
   int arr[100],n,i;
   Printf("Enter the number of elements
   Scanf("%d",&n);
   Printf("\n Enter %d elements:\n\n",n);
   for(i=0;i<n;i++)
   {
     Scanf("%d",&arr[i]);
   }
   quick_Sord(arr,0,n-1);
   Printf("\n The Sorted Array is:\n\n");
   for(i=0;i<n;i++)
   {
     Printf("%4d",arr[i]);
   }
   return 0;
}
void quick_Sort(int arr[], int low, int
{
   int j;
   if(low<high)
   {
     j=Partition(arr,low,high);
```

```
quick_sort(arr, low, j-1);
quick_sort(arr, j+1, high);
}
}
int Partition (int arr[100], int low, int high)
{
    int pivot, j, temp, i, k;
    if(low<high)
    {
      pivot=high;
      i=low;
      j=high;
      while(i<j)
      {
        while((arr[i]<=arr[pivot] && (i<high))
        {
          i++;
        }
      while(arr[j]>arr[pivot])
      {
        j--;
      }
      if(i<j)
      {
        temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
      }
    }
    temp=arr[pivot];
    arr[pivot]=arr[j];
    arr[j]=temp;
}
return 0;
}
```

Merge Sort

ven a list of N array element apply Merge sort. Merge Sort is a Divide and Conquer algorithm. It divide
ay in two halves, calls itself for the two halves and then merges the two sorted halves. The first line c
integer, N, the number of elements in Array. The second line contains N space-separated integers. P
ay as a row of space-separated integers in each iteration.

ps://www.hackerrank.com/contests/17cs1102/challenges/merge-sort-6

gram :-

```
#include <stdio.h>
Void merge_Sort (int ,int);
Void merge_array (int,int,int,int);
int arr[100];
Void main(){
int i,n;
Printf ("\n Enter number of Elements:\n");
Scanf("%d",&n);
Printf ("Enter Elements:\n");
for(i=0;i<n;i++)
Scanf ("%d", &arr[i]);
merge_Sort(0,n-1);
Printf ("\n Sorted Data:");
for(i=0;i<n;i++){
Printf ("%3d", arr[i]);
}
}
Void merge_Sort(int l, int h) {
int m;
if(l<h){
m=(l+h)/2;
merge_Sort(l,m);
merge_Sort(m+1,h);
// Merging two arrays
merge_array(l,m,m+1,h);
}
}
```

```
Void merge_array(int a, int b, int c, int d)
{
    int t[100];
    int i=a, j=c, k=0;
    while(i<=b && j<=d)
    {
        if(arr[i] < arr[j])
            t[k++] = arr[i++];
        else
            t[k++] = arr[j++];
    }
    // collect remaining elements
    while (i<=b)
    t[k++] = arr[i++];
    while (j<=d)
    t[k++] = arr[j++];
    for(i=a, j=0; i<=d; i++, j++)
    arr[i] = t[j];
}
```

ost-lab Task:

. Noor and his pond

loor is doing fish farming. There are N types of fish. Each type of fish has size(S) and eating factor
ith eating factor of E, will eat all the fish of size ≤E. Help Noor to select a set of fish such that the
et is maximized as well as they do not eat each other.

ttps://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-problems/algorithm/noo
ond-760eabe0/

Program :-

```
#include<stdio.h>
void quick_sort(int *a, int first, int last)
{
   int i,j,temp,pivot;
   if(first<last)
   {
      pivot=first;
      i=first;
      j=last;
      {
        while(a[i]<=a[pivot])
        {
          i++;
        }
        while(a[j]>a[pivot])
        j--;
        if(i<j)
        {
          temp=a[i];
          a[i]=a[j];
          a[j]=temp;
        }
      }
      temp=a[j];
      a[j]=a[pivot];
      a[pivot]=temp;
      quicksort(a, first,j-1);
      quicksort(a, j+1, last);
   }
}
```

```
int main()
{
    int n, pivot;
    Scanf("%d", &n);
    int a[n], i, j;
    for(i=0;i<n;i++)
    {
      Scanf("%d", &a[i])
    }
    quicksort(a,0,n-1);
    for(i=0;i<n;i++)
    {
      Printf("%d", a[i]);
    }
}
```

rd game:

friends decided to play a very exciting online card game. At the beginning of this game, each player gets a of cards, in which each card has some strength assigned. After that, each player picks random card from his and they compare strengths of picked cards. The player who picked card with larger strength wins. There is nner in case both players picked cards with equal strength.

friend got a deck with n cards. The i-th his card has strength ai. Second friend got a deck with m cards. -th his card has strength bi.

friend wants to win very much. So he decided to improve his cards. He can increase by 1 the strength of any for 1 dollar. Any card can be improved as many times as he wants. The second friend can't improve his because he doesn't know about this possibility.

is the minimum amount of money which the first player needs to guarantee a victory for himself?

://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-problems/algorithm/card-1-44e9f4e7/

ram :-

```
#include<stdio.h>
Void quick_Sort (int arr[100], int, int);
int Partition (int arr[100], int, int);
int main()
{
  int arr[100], n, i;
  Printf("Enter the number of elements in the Arra
  Scanf("%d", &n);
  Printf("\n Enter %d elements:\n\n", n);
  for (i=0; i<n; i++)
  { scanf("%d", &arr[i]);
  }
  quick_Sort (arr, 0, n-1);
  Printf("\n The Sorted Array is :\n\n");
  for(i=0; i<n; i++)
  {
    Printf("%4d", arr[i]);
  }
  return 0;
}
```

Week-2

Lab Session:

Date of the Session: ___/___/___ **Time of the Session:** ____to____

Prerequisite:

- Divide and Conquer Algorithm
- Sorting
- swapping items in place
- Partitioning of the array.

Pre Lab Task:

1. Chris and Scarlett were playing a block sorting game where Scarlett challenged Chris that he has to so blocks which arranged in random order. And Scarlett puts a restriction that he should not use reference o median and last blocks to sort, and after sorting one block with reference to other block, for next iterat must choose another block as the reference not the same block (random pivot). Now, Chris wants help fro to sort the blocks. He wanted to sort them in a least time. Help him with the least time complexity algorithm.

Program :-

```
#include<stdio.h>
Void quicksort(int *a, int first, int last)
{
    int i,j, temp,pivot;
    if(first<last)
    {
      pivot=first;
       i=first;
       j=last;
      while(i<j)
      { while(a[i]<=a[pivot])
        {i++;
        }
        while(a[j]>a[pivot])
        j--;
        if(i<j)
        {
          temp=a[i];
          a[i]=a[j];
           a[j]=temp;
}}
```

```
    temp=a[j];
    a[i]=a[pivot];
    a[pivot]=temp;
    quicksort(a,first,j-1);
    quicksort(a,j+1,last);
  }
}
int main()
{
  int n,pivot;
  Scanf("%d",&n);
  int a[n],i,j;
  for(i=0;i<n;i++)
  {
    scanf("%d",&a[i])
  }
  quicksort(a,0,n-1);
  for(i=0;i<n;i++)
  {
    Printf("%d",a[i]);
  }
}
```

In Lab Task

1. Implement Shell Sort:

Given a list of N array elements apply shell sort on array.

Input Format

The first line contains the integer N, the size of the array.
The next line contains N space-separated integers.

Constraints

- 1<= N <= 1000
- -1000<= a[i]<= 1000

Output Format

Print the array as a row of space-separated integers in each iteration.

https://www.hackerrank.com/contests/17cs1102/challenges/3b-implement-shell-sort

Program :-

```
#include<stdio.h>
void shell(int a[],int n)
{ int i,j,temp,g,p;
for(g=n/2;g>=1;g=g/2)
{ for(i=g;i<n;i++)
{ for(j=i;j>=g;j=j-g)
{ if(a[j]<a[j-g])
{ temp=a[j];
a[j]=a[j-g];
a[j-g]=temp;
}
}
for(p=0;p<n;p++)
Printf("%d",a[p]);
Printf("\n");
}
}
}
```

```
int main( )
{
  int n;
  scanf("%d",&n);
  int a[n],i;
  for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   shell(a,n);
}
```

3. Insertion Sort

You are given an array AA of size NN. Sort the array using Insertion Sort and print the sorted array.

https://www.codechef.com/DSCA2019/problems/NSECDS03

Program :-

```
#include<stdio.h>
void insrt(int a[],int n)
{ int i,j,k,temp;
  for(i=1;i<n;i++)
   {
     for(j=i;j>0;j--)
     { if(a[j]<a[j-1])
        {
          temp=a[j];
           a[j]=a[j-1];
        a[j-1]=temp;
        }
      for(k=0;k<n;k++)
       Printf("%d ", a[k]);
       Printf("\n");
   }
}
int main()
{
  int n,i,m;
  Scanf("%d",&n);
  int a[n];
  for(i=0;i<n;i++)
     Scanf("%d",&a[i]);
  for(m=0;m<n;m++)
     Printf("%d ", a[m]);
  Printf("\n");
  insrt(a,n);
}
```

Quick sort 1 – Partition

previous challenges covered Insertion Sort, which is a simple and intuitive sorting algorithm with a running . In these next few challenges, we're covering a *divide-and-conquer* algorithm called Quick sort (also known *Partition Sort*). This challenge is a modified version of the algorithm that only addresses partitioning. It is lemented as follows:

s://www.hackerrank.com/challenges/quicksort1/problem

gram :-

```
#include<stdio.h>
Void quick_Sort (int [], int ,int);
int Partition (int[], int, int);
int main( )
{
   int n,i;
   Scanf("%d",&n);
   int a[n];
   for(i=0; i<n;i++)
   Scanf ("%d", &a[i]);
   quick_Sort (a,0,n-1);
   for(i=0;i<n;i++)
   Printf ("%d ",a[i]);
   return 0;
}
Void quick_Sort(int a[], int low,int high)
{
   int j;
   if(low<high)
   {
      j= Partition(a, low, high);
      quick_Sort (a,low,j-1);
      quick_Sort (a,j+1,high);
   }
}
int Partition(int a[],int low, int high)
{
   int Pivot, j, temp, k, i;
   if(low<high)
   { Pivot=high;
```

```
while((a[i]<=a[pivot])&&(i<high))
{
  i++;
}
while(a[j]>a[pivot])
{
  j--;
}
if(i<j)
{
  temp=a[i];
  a[i]=a[j];
  a[j]=temp;
}
}
temp=a[pivot];
a[pivot]=a[j];
a[j]=temp;
}
return j;
}
```

st Lab Task:

Missing Number

ou are given an array *A*. You can decrement any element of the array by *1*. This operation can be repeated ny number of times. A number is said to be missing if it is the smallest positive number which is a multiple f *2* that is not present in the array *A*. You have to find the maximum missing number after all possible ecrements of the elements.

ttps://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-roblems/algorithm/biggest-cake-possible-6d5915e7/

rogram :-

```
#include<stdio.h>
void quicksort(int *a,int first, int last)
{
  int i,j,pivot,temp;
  if(first<last)
  {
    pivot=first;
      i=first;
    j=last;
   while(i<j)
   {
     while(a[i]<=a[pivot])
     {i++;
     }
     while(a[j]>a[pivot])
     j--;
     if(i<j)
     {temp=a[i];
       a[i]=a[j];
       a[j]=temp;
     }
   }
   temp=a[j];
   a[j]:a[pivot];
   a[pivot]=temp;
   quicksort(a,first,j-1);
   quicksort(a,last,j+1);
}
```

```
}
int main()
{
    int n, pivot;
    scanf("%d", &n);
    int a[n], i, j;
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    quicksort(a, 0, n-1);
    for(i=0; i<n; i++)
    {
        printf("%d", a[i]);
    }
}
```

2. One-Sized Game:

Ladia and **Kushagra** are playing the **One-Sized Game**. The rules of this game are pretty simple. During the game's play, if at any point of time, the array is left with only one element, Ladia wins the game otherwise Kushagra wins.

Given an array consisting of N elements, print whether Ladia will win or Kushagra.

https://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-problems/algorithm/one-sized-game/

Program :-

```
#include<stdio.h>
void quicksort(int *a, int first, int last)
{
    int i,j, temp,pivot;
    if(first<last)
    {
      pivot=first;
      i=first;
      j=last;
      while(i<j)
      {
        while(a[i]<=a[pivot])
        {i++;
        }
      while (a[j]> a[pivot])
      j--;
      if(i<j)
      {temp=a[i];
       a[i]= a[j];
       a[j]=temp;
      }
    }
     temp=a[j];
     a[j]=a[pivot];
     a[pivot]=temp;
```

```
    quicksort(a, first, j-1);
    quicksort(a, last, j+1);
  }
}
int main()
{
  int n, pivot;
  scanf("%d", &n);
  int a[n], i, j;
  for(i=0; i<n; i++)
  {
    scanf("%d", &a[i]);
  }
  quicksort(a, 0, n-1);
  for(i=0; i<n; i++)
  {
    Printf("%d", a[i]);
  }
}
```

Week-3

Lab Session:

Date of the Session: ___/___/___ **Time of the Session:** ____to____

Prerequisite:

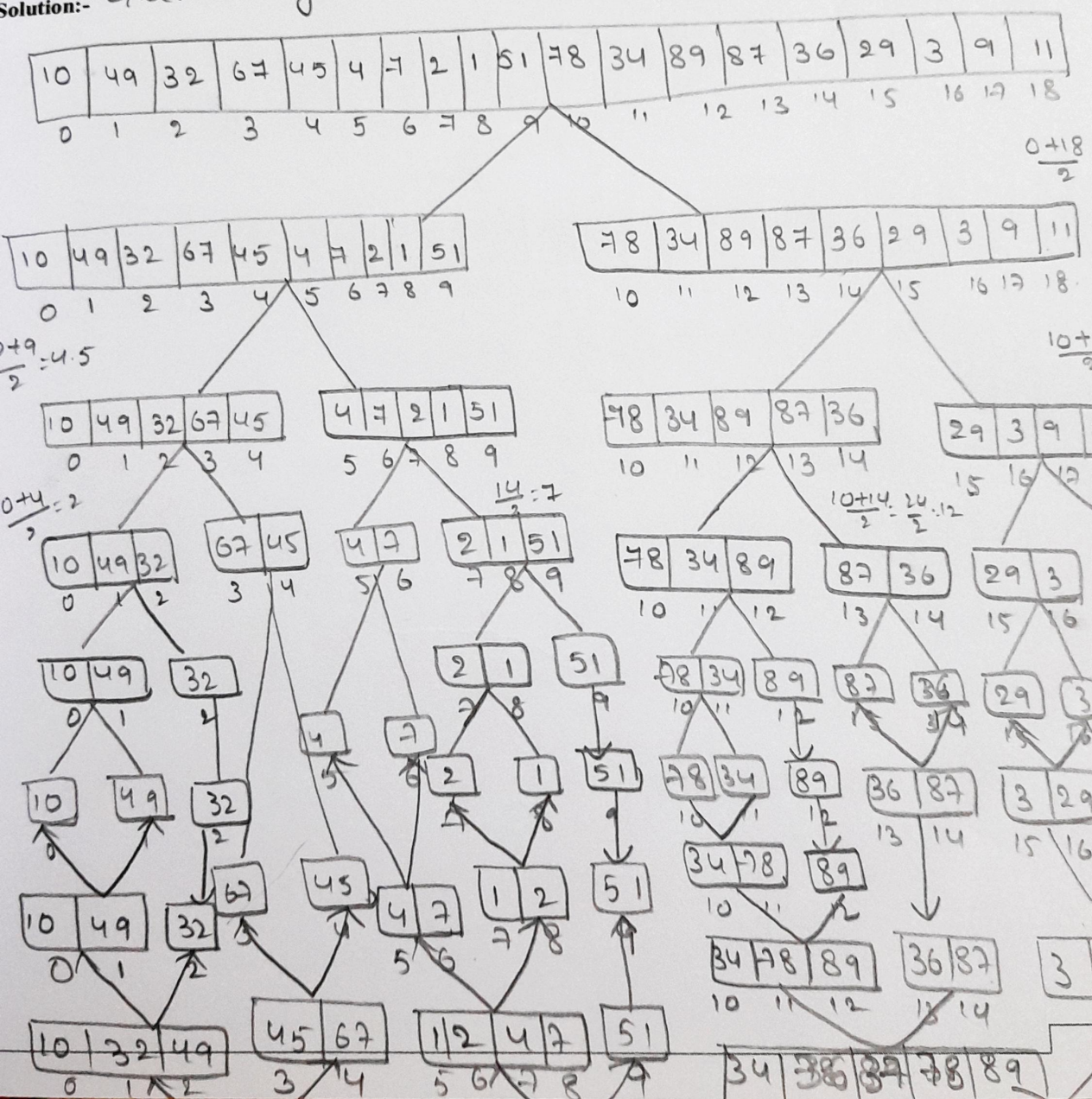
- Divide and Conquer Algorithm
- Sorting
- swapping items in place
- Partitioning of the array.

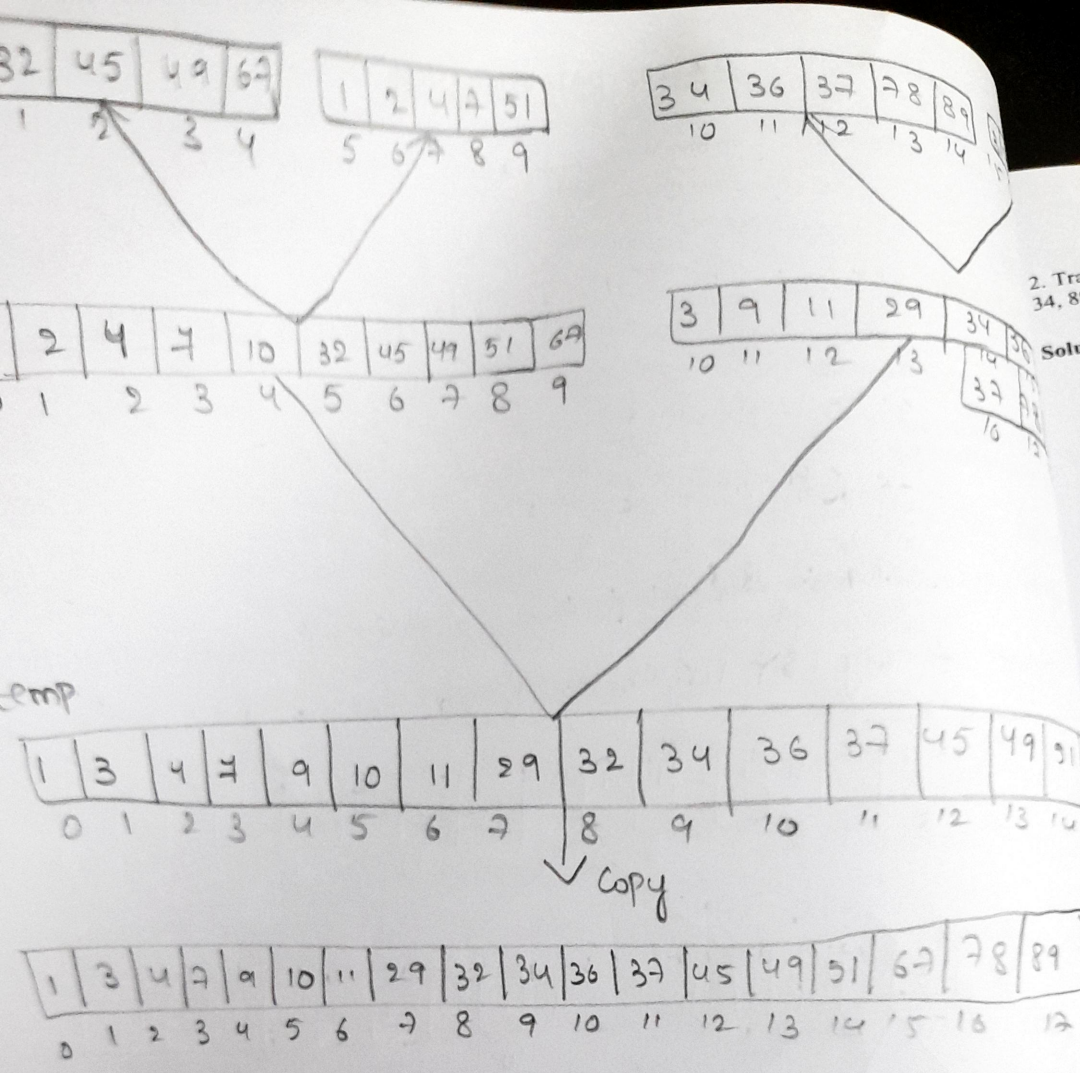
Pre Lab Task

1. Trace out the output of the following using Merge sort.10, 49, 32, 67, 45, 4, 7, 2, 1, 51, 78, 34, 89, 87, 36, 29, 3, 9, 11.

Solution:- Given Array is :- 10, 49, 32, 67, 45, 4, 7, 2, 1, 51, 78, 34, 89, 87, 36, 29, 3, 9, 11

n = 19




32 45 49 67
1 2 3 4
1 2 4 7 51
5 6 7 8 9
34 36 37 78 89
10 11 12 13 14
2 4 7 10 32 45 49 51 69
1 2 3 4 5 6 7 8 9
3 9 11 29 34
10 11 12 13
37
14
temp
1 3 4 7 9 10 11 29 32 34 36 37 45 49 51
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
copy
1 3 4 7 9 10 11 29 32 34 36 37 45 49 51 67 78 89
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

2. Trace the insertion sort algorithm to arrange the following data in ascending order 49, 32, 67, 45, 4, 2, 1, 78, 34, 89, 87.

Solution:-

Example :-

49	32	67	45	4	2	1	78	34	89	87

Pass 1 :- 49 32 67 45 4 2 1 78 34 89 87

32 49 67 45 4 2 1 78 34 89 87

Pass 2 :- 32 49 67 45 4 2 1 78 34 89 87

32 49 45 67 4 2 1 78 34 89 87

32 45 49 67 4 2 1 78 34 89 87

Pass 3 :- 32 45 49 4 67 21 78 34 89 87

32 45 4 49 67 2 1 78 34 89 87

32 4 45 49 67 2 1 78 34 89 87

4 32 45 49 67 2 1 78 34 89 87

Pass 4 :- 4 32 45 49 2 67 1 78 34 89 87

4 32 45 2 49 67 1 78 34 89 87

4 32 2 45 49 67 1 78 34 89 87

4 2 32 45 49 67 1 78 34 89 87

2 4 32 45 49 67 1 78 34 89 87

Pass 5 :- 2 4 32 45 49 1 67 78 34 89 87

2 4 32 45 1 49 67 78 34 89 87

2 4 32 1 45 49 67 78 34 89 87

2 4 1 32 45 49 67 78 34 89 87

2 1 4 32 45 49 67 78 34 89 87

1 2 4 32 45 49 67 78 34 89 87

Pass 6:- 1 2 4 32 45 49 67 34 78 89 87

1 2 4 32 45 49 34 67 78 89 87

1 2 4 32 45 34 49 67 78 89 87

1 2 4 32 34 45 49 67 78 89 87

Pass 7:- 1 2 4 32 34 45 49 67 78 89 87

Pass 8:- 1 2 4 32 34 45 49 67 78 89 87

Pass 9:- 1 2 4 32 34 45 49 67 78 89 87

1 2 4 32 34 45 49 67 78 87 89

Sorted array is:-

1	2	4	32	34	45	49	67	78	87	89
0	1	2	3	4	5	6	7	8	9	10

In Lab Task

1. Quick Sort

You are given an array AA of size NN. Sort the array using Quick Sort and print the sorted array.

https://www.codechef.com/DSCA2019/problems/NSECDS06/

Program :-

```
#include<stdio.h>
int Void quick_sort(int *a, int first ,int last)
 {
    int i,j, pivot,temp;
    if (first < last)
    {
       Pivot=first;
        i = first;
        j = last;
        while (i<j)
        {
          while (a[i]<=a[pivot])
          {
            i++;
          }
          while (a[j]>a[pivot])
          j--;
          {
            temp=a[i];
             a[i]=a[j];
              a[j]=temp;
          }
       }
        temp=a[j];
        a[j] = a[pivot];
        a[pivot]= temp;
        quicksort (a,j-1, first);
        quicksort (a,j+1, last);
    }
}
```

```
int main()
{
    int n, pivot;
    scanf("%d", &n);
    int a[n], i, j;
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    quicksort(a, 0, n-1);
    for(i=0; i<n; i++)
    {
        Printf("%d", a[i]);
    }
}
```

2. Max Power

Given an array A having N distinct integers.

The power of the array is defined as:

- max(A[i]−A[j]) where 2≤i≤N
- for each *i*, *j* is the largest index less than *i* such that A[j]<A[i].

Let's say the array is {1,2,5}, then the power of the array is max((2−1),(5−2)), which simplifie to max(1,3) which is equal to *3*.

Operation Allowed:

If you are allowed to choose any two indices *x* and *y* and swap A[x] and A[y], find out the maximum power t can be achieved.

https://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-problems/algorithm/increasing-subsequence-fbb63e3c/

Program :-

```
#include<stdio.h>
void quicksort(int *a, int first, int last)
{
   int i,j,temp,pivot;
   if(first<last)
   {
     pivot=first;
      i=first;
      j=last;
     while(i<j)
     {  while(a[i]<=a[pivot])
      {  i++;
      }
      while (a[j]>a[pivot])
      j--;
     if(i<j)
     {
       temp=a[i];
        a[i]=a[j];
        a[j]=temp;
  }  }
  temp=a[i];
  a[i]=a[pivot];
  a[pivot]=temp;
```

```
    quicksort(a,first,j-1);
    quicksort(a,last,j+1);
  }
 }
int main()
{
  int n,pivot;
  scanf("%d",&n);
  int a[n],i,j;
  for(i=0;i<n;i++)
  {
    scanf("%d",&a[i]);
  }
  quicksort(a,0,n-1);
  for(i=0;i<n;i++)
  {
    printf("%d",a[i]);
  }
}
```

u are the king of Pennsville where you have 2N workers. All workers will be grouped in association
e 2,so a total of *N* associations have to be formed. The building speed of the ith worker is Ai. To make
ociation, you pick up *2* workers. Let the minimum building speed between both workers be *x*, then
ociation has the resultant building speed *x*.

u have to print the maximum value possible of the sum of building speeds of *N* associations if you make
ociations optimally.
s://www.hackerearth.com/practice/algorithms/sorting/merge-sort/practice-
blems/algorithm/maximum-sum-of-building-speed-00ab8996/

gram :-

```
#include <stdio.h>
Void merge_Sort (int,int);
Void merge_array (int,int, int,int);
int arr[100];
void main(){
int i,n;
Printf("\n Enter number of Elements :\n");
Scanf("%d",&n);
Printf("Enter Elements:\n");
for(i=0;i<n;i++)
 Scanf("%d",&arr[i]);
 merge_Sort(0,n-1);
Printf("\n Sorted Data :\n");
for(i=0;i<n;i++){
 Printf("%3d", arr[i]);
}
}
void merge_ Sort(int l, int h){
 int m;
    if(l<h){
      m=(l+h)/2;
      merge_Sort(l,m);
      merge_sort(h,m+1);
      //merging two arrays
```

```
        merge_array(l, m, m+1, h);
    }
}
void merge_array(int a, int b, int c, int d)
{
    int t[100];
    int i=a, j=c, k=0;
    while(i<=b && j<=d) {
        if(arr[i]<arr[j])
            t[k++]=arr[i++];
        else
            t[k++]=arr[j++];
    }
    //collect remaining elements
    while(i<=b)
        t[k++]=arr[i++];
    while (j<=d)
        t[k++]=arr[j++];
    for(i=0, j=0; i<=d; i++, j++)
        arr[i]=t[j];
}
```

Post Lab Task:

1. Benny and Gifts:

Little pig Benny has just taken a shower. Now she is going to buy some gifts for her relatives. But the problem that Benny doesn't know how to reach to the gift shop. Her friend Mike has created a special set of instructi for her. A set of instructions is a string which consists of letters {'L', 'R', 'U', 'D'}.

For simplicity, let's assume that Benny is staying at point (0, 0) on the infinite plane. She consistently fulfill the instructions written by Mike. Let's assume that now she is staying at point (X, Y). Then depending on wh the current instruction she moves in some direction:

- 'L' -- from (X, Y) moves to point (X, Y - 1)
- 'R' -- from (X, Y) moves to point (X, Y + 1)
- 'U' -- from (X, Y) moves to point (X - 1, Y)
- 'D' -- from (X, Y) moves to point (X + 1, Y)

The weather is cold because it's winter now. Initially, all points are snowy. But if Benny have already some point at any time this point becomes icy (because Benny has just taken a shower). Every time, whe makes a step into icy points she slips and falls down.

You are given a string S which denotes a set of instructions for Benny. Your task is to calculate how m she will fall down.

https://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-problems/algorithm and-gifts-marcheasy-3/

Program :-

```
#include<stdio.h>
void quicksort (int *a , int first ,int last)
{
   int i,j, temp, pivot;
   if (first < last)
   {
      pivot= first;
      i = first;
      j= last;
      while(i<j)
      {
        while (a[i]<=a[pivot])
        { i++;
        }
        while[a[j] > a(pivot])
        j--;
        if(i<j)
```

```
    {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
    }
}
    temp=a[j];
    a[j]=a[pivot];
    a[pivot]=temp;
    quicksort(a,first,j-1);
    quicksort(a,j+1,last);
  }
}
int main()
{
  int n,pivot;
  scanf("%d",&n);
  int a[n],i,j;
  for(i=0;i<n;i++)
  {
    scanf("%d",&a[i]);
  }
  quicksort(a,0,n-1);
  for(i=0;i<n;i++)
  {
    Printf("%d",a[i]);
  }
}
```

arth and The Meteorites

upon a time, the Earth was a flat rectangular landmass. And there was no life. It was then that the sky lit up meteorites falling from out of space. Wherever they fell on the planet, a river was born, which flowed in all ections (North, East, West, South), till the waters reached the edge of the Earth and simply fell off into .

these rivers criss-crossed and divided the one huge landmass (Pangaea) into many smaller landmasses. the lifeless (there was no life, remember?), want to know the number of landmasses on the planet after all eteorites have fallen. They also want to know the area of the smallest and largest landmass. Can you help eless in this question?

://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-ems/algorithm/earthandthemeteorites-qualifier2/

ram :-

```
#include<stdio.h>
void quicksort(int *a,int first,int last)
{
  int i,j,temp,pivot;
  if(first<last)
  {
    pivot=first;
    i=first;
    j=last;
    while(i<j)
    {
      while(a[i]<=a[pivot])
      {
        i++;
      }
      while(a[j]>a[pivot])
      j++;
      if(i<j)
        {
          temp=a[i];
          a[i]=a[j];
          a[j]=temp;
        }
    }
    temp=a[j];
    a[j]=a[pivot];
    a[pivot]=temp;
```

```
        quicksort(a,j-1,first);
        quicksort(a,j+1,last);
    }
}
int main()
{
    int n,pivot;
    scanf("%d",&n);
    int a[n],i,j;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    quicksort(a,0,n-1);
    for(i=0;i<n;i++)
    {
        printf("%d", a[i]);
    }
}
```