Practice - 1

1) Declare an array of 5 subjects accepts subject marks from user calculate the avg or percentage

→
```
#include <stdio.h>
int main ()
{
  int i, marks[5], sum = 0;
printf ("Enter marks of 5 subjects :");
 for (i = 0; i < 5; i++)
  {
     scanf ("%d", &marks[i]);
  }
   for (i = 0 ; i<5; i++)
  {
     sum += marks[i];
  }

float avg = sum/5;
printf ("Average = %.2f", Avg);

return 0;
}
```

O/P :- enter mar of 5 sub
Sub-1: 55
Sub 2: 65
Sub 3: 59
Sub 4: 85
Sub 5: 98

Average mar = 72.400002

2) Write a program to accept number n from user and check if it is even or odd

⇒

```
# include <stdio.h>

int main () {
int a;
printf ("Enter a number:");
scanf ("%d", &a);

if (a % 2 == 0) {
   printf ("even");
}

else {
      printf ("odd");

}

   return 0;

}
```

O/P: Enter a number: 14
even

3 Write a program to print prime no. between 1 to 50

⇒
```
# include <stdio.h>
int main () {
int i, j;
printf ("Prime no. from 1 to 50 are: \n");

for (i = 2; i<=50; i++) {
  int isprime = 1;
for (j=2; j<i; j++) {
if (i % j == =0) {
  isprime = 0;
    break;
  }
}
if (isprime) {
  printf ("%d", i)
}
}
  return 0;
}
```

O/p:- Prime numbers from 1 to 50 are:-
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

4) Write a menu driven program.

⇒

```
#include <stdio.h>
int main() {
  int day;

printf("Enter day number (1 for Monday,
        7 for Sunday): ");
scanf("%d", &day);

  printf("It is a holiday! \n");
    break;
case 1:
case 2:
case 3:
case 4:
case 5:

printf("It is a working day. \n");

default;

printf("Invalid day number! Please
        enter 1 to 7. \n")

}
  return 0;
}
```

O/P:-
Enter day number = 5
It's a working day

output:-

```
Enter a number: 121
121 is a palindrome
```

5. Write a program to accept number from user and check it its palindrom or not

⇒

```
# include <stdio.h>
int main ()
{
    int num, reversed = 0, remainder,
    orignal;

printf ("Enter an in integer: ");
scanf ("%d", & num);

orignal = num;
while (num != 0)
{
    remainder = num % 10;
    reversed = reversed * 10 + remainde
    num = num / 10;
}
if (orignal == reversed)
{
    printf ("%d", is a palindrome", orignal);
}
else
{
    printf ("%d is not palindoome", orignd)
}
return 0;
}
```

Experiment 1

WAP to implement array operation

1) Find avg of 10 no. using array

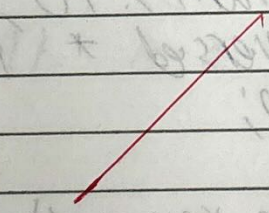
2) Display the following pattern

```
*
# #
* * *
# # # #
```

3) Find first repeating element of array

4) Find the greatest and smallest element in array

5) WAP squaring the odd position element



)

⇒
```
# include <stdio.h>

int main () {
int number [10] , i , sum=0
float average;

printf ("Enter 10 numbers : \n");

for (i=0; i<10; i++) {

    scanf ("%d", & numbers [i]);
    sum = sum + numbers [i];

}

average = sum / 10.0;

printf ("Average = %.2f\n", average);

return 0;

}
```

O/P ⇒
```
Enter 10 no.!-
8, 9, 7, 6, 5, 9, 4, 8, 7, 2
Average = 6.50
```

2)

```
#include <stdio.h>

int main() {
  int rows = 4;

for (int i = 1; i <= rows; i++){
  for (int j = 1; j <= i; j++){
    if (j % 2 == 0)
      printf ("#");

  else
      printf ("*");

  printf ("\n");

  }

  return 0;

  }
```

O/P ⇒ Output of (3)

Enter 5 elements : 5

4

5

8

4

First element repeating is 5

```
# include <stdio.h>
3) int main () {
   int arr [100], n, i, j, found;

   printf ("Enter number of elements: ");
   scanf ("%d", &n);
   printf ("Enter %d element: \n", n)
   for (i = 0; i<n; i++) {
    scanf ("%d", &arr[i]);
   }
     for (i = 0; i <n; i++) {
     for (j = i+1; j <n; j++) {
     if (arr[i] == arr[j] {
   printf ("first repeating element is %d\n", arr[i]);
   printf ("No repeating elements found\n");
    return 0;
     }
     }
     }
   printf ("\no repeating element found. \n");
    return 0;
     }
```

O/P Output of (2)

```
   *
  # #
 * * *
# # # #
```

4)
```
#include <stdio.h>
int main () {
  int n;
printf("Enter the size of the array:");
 scanf("%d", &n);
 int arr[n];
printf("Enter %d element:\n", n);
  for (int i=0; i<n; i++) {
   scanf("%d", &arr[i]);
  }
   int max = arr[0];
   int min = arr[0];

for (int i=1; i<n; i++) {
  if (arr[i] > max)
     max = arr[i];
  if (arr[i] < min)
     min = arr[i];
  }
printf("Grates element = %d\n", max);
printf("Smallest element = %d\n", min);

 return 0; }
```

O/p

```
Enter 5 element :- 5
                   8
                   7
                   9
                   4
Greatest element = 9
Smallest element = 4
```

4.
```
#include <stdio.h>
int main () {
    int n;
printf ("enter the size of the array: ");
 scanf ("%d", &n)

 int arr [n];
printf ("enter %d elements: \n", n);
 for (int i=0; i<n; i++) {
   scanf ("%d", &arr[i]);
   }

 for (int i=0; i<n; i++) {
     if (i % 2 != 0) {
     arr[i] = arr [i] * arr[i];
     }
     }
printf ("Updated array (after squaring odd-position
      elements): \n");
 for (int i=0; i<n; i++) {
   printf ("%d", arr [i]);
 }
  return 0; }
```

O/p
```
enter the size of the array: 5
enter 5 elements:
5
4
9
5
7
Updated array (after squaring odd-position elements):
5, 16, 9, 64, 7
```

11/7/25

Practical - 2

1) Search Data using linear search Consider following list to print 56, 36, 84, 57, 1, 0, 67, 64

a) Search item from list and write element found with procedure
b) Also no. 55

⇒
```
#include <stdio.h>
int main ()
{
  int a[8], k
printf ("Enter 8 elements:");
 for (int i=0; i<8; i++)
  {
    scanf ("%d", &a[i]);
  }
printf ("Enter the search key:");
 scanf ("%d", &k);

 for (int i = 0, i<8; i++)
   {
     if (a[i] == k)
   {
     printf ("Element found");
      return 0;
   }
 }
printf ("\n Enlement not found");
return 0; }
```

⇒ a Output

Enter 8 elements : 56
36
89
57
1
0
67
9

Enter the search key : 9
element found at 5

b output

Enter 8 elements : 56
36
89
57
1
0
67
9
Enter the search key : 55

element not found

2. Search data using binary search

```
#include <stdio.h>
int main()
{
  int a[10], i, l, h, m, k;
printf("enter 10 elements (sorted)");
 for (int i=0; i<=10; i++)
  {
    scanf("%d", &a[i]);
  }
printf("enter the search key:");
 scanf("%d", &k);
 l=0, h=9;
 while (l<=h)
  {
   m = (l+h)/2;
  if (a[m] == k)
  {
  printf("element found
   break;
  }
  else
  {
    if (k<a[m])
   {
     h=m-1;
   }
   else
   {
     l = m+1;
   }
```

```
    }
  }
  printf("\n element not found");
return 0;
}
```

```
Enter 10 elements :
1
2
3
4
5
6
7
8
9
10

enter the search key: 5
element found at index: 4
```

Q3) Compare linear & binary search!

	Linear Search	Binary Search
1.	It works on both sorted & unsorted arrays.	It works only on sorted array
2.	Time complexity in (n)	Time complexity in (log n)
4.	Simple to implemet	Complex to implemet
5.	Compares with each 1 first element	Compared with middle element.
5.	Algorithm type is sequential	Algorithm type is divide & conaver

Q4) State limitations of linear search in terms or time complexity

⇒ linear search has limitations in terms or time complexity especially for large datas. Its worst case and average case time complexity is (n) meaning it may need to scan every element in array to determine if it is present or not, which makes it slower. Tu binary search is on sorted data. Hence performace is poor if data size grows.

Practice question.

1. Write a program to copy the elements of one array into another array in a reverse order.

⟹
```
# include <stdio.h>
int main ()
{
    int original[100], reversed[100]
    int n;
printf ("\n Enter the number of
            elements in array:");
scanf ("%d" &n);
printf ("\n Enter %d elements ", n);

    for (int i = 0; i<n; i++)
    {
        scanf ("%d", &original[i]);
    }
    for (int i = 0; i<n; i++)
    {
        reversed [i] = original [n-1-i];
    }

printf ("\n Reversed array: ");

for (int i = 0; i<n; i++)
```

```
{
    printf ("%d", reversed [i]);
}
    return 0;
}
```

Output :-

Enter number of elements in array : 4

Enter 4 elements : 8 5 9 3

Reversed array : 3 9 5 8

2. WAP in C to count total no. of duplicate elements in an array

⇒
```
#include <stdio.h>
int main()
{
  int a[100], n, i, j, count;
printf("\n Enter number of elements:
  scanf("%d", &n);
printf("\n Enter %d elements", n)
  for(i=0; i<n; i++)
  {
    scanf("%d", &a[i]);
  }
   for(i=0; i<n; i++)
   {
     for(j=i++; j<n; j++)
   {
     if(a[i]==a[j])
   {
      count++;
     break;
   }
   }
   }

printf("\n Total Duplicate elemats:
          %d: count);

return 0;
}
```

Output;

```
Enter number of elements : 4
Enter 4 elements : 5 5 4 7
Total Duplicate elements : 1
```

3. WAP in C to print all unique element in array.

⇒
```
# include <stdio.h>
int main ()
{
  int n, j, i, count;

printf ("\n Enter number of elements : ");
 scanf ("%d", &n)
printf ("\n Enter %d elements : ", n);
  for (i=0; i<n ; i++)
  {
    scanf ("%d", &a [i]);
  }
printf ("\n unique elements are : ");
  for( i=0 ; i<n ; i++)
  {
    count = 0;
  for (j=0; j<n ; j++)
{
  if (a [i] == a[j])
  {
    count ++;
  }
  }
```

```
if (count == 1)
{
    printf ("%d ", a[i]);
}
}
return 0;
}
```

Output:

```
Enter number of elements: 4
Enter 4 elements: 5 5 8 7
Unique elements are: 8 7
```

4. WAP to seperate odd and even integers into separate arrays.

⇒
```
#include <stdio.h>
int main()
{
  int a[100], even[100], odd[100];
  int n, i, e=0, o=0;

printf("\n Enter number of elements:");
 scanf("%d", &n);

printf("\n Enter %d elements :", n);
   for(i=0; i<n; i++)
   {
      scanf("%d", &a[i]);
   }
 for(i=0; i<n; i++)
 {
  if(a[i]%2==0)
  {
     even[e] = a[i];
      e++;
  }
   else
   {
     odd[0] = a[i]
      o++;
   }
 }
```

```
printf ("Even numbers : ");
   for (i=0; i<e ; i++)
    {
      printf ("%d", even[i]);
    }

printf ("Odd Numbers : ");
   for (i=0; i<o; i++)

  {
    printf ("%d", odd[i]);

  }

  return 0;

  }
```

Output

```
Enter number of elements :6
Enter 6 element : 5 8 9 7 2 1
Even numbers = 8 2
Odd numbers = 5 9 7 1
```

5) Write a C program to find second smallest element in an array

⇒
```
# include <stdio.h>
int main ()
{
   int arr [] = { 10, 5, 8, 20, 23}
int n;
int i, j, temp;
for ( i = 0; i < n ; i++)
   {
      for (j = i+1 ; j < n ; j++)
      {
         if ( arr [i] > arr[j] )
      {
         temp = arr [i]
         arr [i]  = arr [j]
         arr [j]  = temp
      }
   }
}

for (i=1; i < n ; i++)
   {
   if (arr [i] ! = arr [0])
   {
   printf ("Second smallest element is:
            %d", arr [i]);

return 0;
   }
}
```

6) WAP to count total number of words in a string.

⇒
```
#include <stdio.h>
int main()
{
  char str[100];
int i, words = 1;
printf("\n Enter a string:");
 scanf("%[^\n]", str);
for(i=0; str[i]!='\0'; i++)
 {
   if (str[i] == ' ')
   {
     words++;
   }
 }
printf("Total number of words:
                %d", words);

return 0;
 }
```

Output:

Enter a string: I study Engineering
Total no. of words: 3

Experiment - 3

1. Write 'C' program to sort data from given array using Bubble sort & selection sort

. Sort elemats in ascending order using Bubble sort.

⇒
```
# include <stdio.h>
int main ()
{
  int a[50], n, i, j, temp;
printf("\n Enter the numbe of
                element : ");
scanf ("%d", &n);

printf ("\n Enter %d " elements : "n);
   for (i=0; i<n; i++)
    {
     scanf ("%d"; & a[i]);
    }
   for( i=0; i<n-1; i++)
   {
     for (j = 0; j<n-i-1; j++)
   {
     if (a[j] >a [j+1])
   {
     tem = a[j];
   a[j] = a[j+1];
```

```
        a[j+1] = temp;
      }
    }
  }
printf("\n sorted array in ascending
                order:");
for(i=0; i<n; i++)
  {
    printf("%d", a[i]);
  }
    return 0;
  }
```

Output :-

```
enter no. of elements: 5
Enter 5 elements: 8 6 69 45 1
Sorted Array in ascending order
1 6 8 45 69
```

ii Sort elements in descending order using selection sort

⇒
```
#include <stdio.h>
int main()
{
    int a[50], n, i, j, max, temp
printf("\n enter number of elements");
 scanf("%d", &n)

printf("\n enter %d elements: ", n);
  for (i=0; i<n; i++)
{
   scanf("%d", &a[i]);
  }
for (i=0; i<n-1; i++)
  {
   max = i;
for (j=1+i; ; j<n; j++)
  {
    if (a[j] > a[max])
  {
     max = j;
   }
   }
  temp = a[i];
  a[i] = a[max];
  a[max] = temp;

  }
```

```
printf ("\n Sorted array in descending
                order :");

for (i=0; i<n; i++)
  {
  printf ("%d", a[i]);
  }

return 0;

}
```

Output :-

```
Enter no. of elements : 5
Enter 5 elements : 8 6 3 7 8 9
Sorted array in descending order
98    78    8    6    3
```

iii) Find the number of comparisons required in bubble sort method of the following list having 5 numbers.

Pass 1) 100 200 300 400 500

100 200 300 400 500

100 200 300 400 500

100 200 300 400 500

100 200 300 400 500

The list is already sorted

iv. Sort the given array in ascen order using selection sort method and show diagrammatic represtation of every its ration

loop: 500, -20, 30, 14, 50

Pass 1) 500 -20 30 14 50

-20 500 30 14 50

-20 500 30 14 50

-20 500 30 14 50

-20 500 30 14 50

Pass 2 -20 500 30 14 50

-20 30 500 14 50

-20 14 500 14 50

-20 14 500 14 50

Pass 4) -20 14 500 30 50

-20 14 30 500 50

-20 14 30 500 50

classmate

Date
Page

Pass 3) -20 14 30 500 50

-20 14 30 50 500

Nawal 8/9/25

Experiment 4

Q Write 'C' program to sort data from given array using insertion sort & Radix sort.

1. Sort elements in ascending order using insertion sort

2. Sort elements in ascending order using radix sort

3. What is the output of inserting sort after second iteration given the following sequence of numbers:
   7, 3, 5, 1, 9, 8, 4, 6

4. Sort the following numbers using radix sort.
   1) 100, 225, 390, 4130, 956, 99, 5431
   2) 25, 6, 99, 145, 239, 20, 18

5. Sort the following elements using heap
   1) 6, 5, 9, 2, 1, 4
   2) 3, 2, 1, 10, 4

6. Sort the following using shell sort
   1) 22, 18, 29, 7, 9, 55, 21, 8
   2) 3, 10, 15, 12, 1, 5, 2, 6

Ans)1.

```
#include <stdio.h>
int main() {
    int n, i, j, key;
printf("Enter the number of element:");
scanf("%d", &n)
int arr[n]
printf("Enter %d integer : \n", n);
 for(i=0 ; i<n, i++)
  {
     scanf("%d", & arr[i]); }
for(i=1; i<n; i++){
     Key = arr[i];
for(j=i-1; j>=0; j--) {
  if(arr[j] > key) {
  arr[j+1] = arr[j]; }
   else {break;}
    }
   arr[j+1] = key;
printf("Array after pass %d: \n", i);
 for(j=0, j<n; j++) {
  printf("%d", arr[j+1]; } printf["\n"]
   printf("Array sorted in ascending orders
         using insertion sort: \n")
for(i=0; i<n; i++)
  { printf("%d", arr[i]); }

  return 0;

  }
```

Output :-

Enter the number of elements : 5
Enter 5 integers:
3 5 9 1 2
Array after pass 1:
3 5 9 1 2

Array after pass 2

3 5 9 1 2

Array after pass 3

1 3 5 9 2

Array after pass 4

1 2 3 5 9

Array sorted in ascending order using Insertion sort:
1 2 3 5 9

A 2]

```
// Radix sort
#include <stdio.h>
int main() {
int n, int arr[100];
printf("Enter the number of elements (max 100): ");
scanf("%d", &n);
printf("Enter %d integers: \n", n);
for(int i=0; i<n; i++)
{
   scanf("%d", &arr[i]); }
int max = arr[0];
for(int i=1; i<n; i++)
{
   if (arr[i] > max)
{
   max = arr[i]; } }
for(int exp=1; max/exp>0; exp=10){
int output[100] int count[10] = {0};
for(int i=0; i<10; i++) { count[(arr[i]/exp)%10]++; }
for(int i=1; i<10; i++){ count[i] += count[i-1];}
for(int i=n-1; i>=0; i++) {
   output[ count[(arr[i]/exp)%10] = arr[i];
   count[(arr[i]/exp)%10]--;
  }
for(int i=0; i<n; i++) { arr[i] = output[i];}
printf("After pass:");
for(int i=0; i<n; i++) { printf("%d", arr[i]);}
printf("\n");}
printf("Sorted array:");
for(int i=0; i<n; i++){ printf("%d", arr[i]);}
printf("\n");
 return 0;
}
```

Output

Enter the number of elements (max 100): 5
Enter 5 integers:
274 891 3 444 67

After pass: 891 3 274 444 67

After pass: 3 444 67 274 891

After pass: 3 67 274 444 891

Sorted array: 3 67 274 444 891

A3] 7, 3, 5, 1, 9, 8, 4, 6 → Array

First iteration

7, 3, 5, 1, 9, 8, 4, 6

3, 7, 5, 1, 9, 8, 4, 6

Second iteration

3, 7, 5, 1, 9, 8, 4, 6

3, 7, 5, 1, 9, 8, 4, 6

⇒ 3, 5, 7, 1, 9, 8, 4, 6

= output after second iteration for insertion sort

A4] Radix Sort.

a. Arr = 100, 225, 390, 4130, 956, 99, 5431

	0	1	2	3	4	5	6	7	8	9
100	100									
225						225				
390	390									
4130	4130									
956							956			
99										99
5431		5431								

Arr = 100, 390, 4130, 5431, 225, 956, 99

	0	1	2	3	4	5	6	7	8	9
100	100									
340										340
4130				4130						
5131				5131						
225			225							
956						956				
99										99

Arr = 100, 225, 4130, 5431, 956, 340, 99

	0	1	2	3	4	5	6	7	8	9
100		100								
225			225							
4130		4130								
5431						5431				
956										956
340				340						
099	099									

Arr = 99, 100, 4130, 225, 340, 5431, 956

	0	1	2	3	4	5	6	7	8	9
0099	99									
0100	100									
4130					4130					
0225	225									
0340	340									
5431						5431				
0956	956									

Arr = 99, 100, 225, 340, 956, 4130, 5431

The arr is sorted

b→ 25, 6, 99, 145, 239, 20, 18

	0	1	2	3	4	5	6	7	8	9
25						25				
6							6			
99										99
145						145				
239										239
20	20									
18									18	

arr = 20, 25, 145, 6, 18, 99, 239

	0	1	2	3	4	5	6	7	8	9
20			20							
25			25							
145					145					
06	06									
18		18								
99										99
239				239						

arr - 6, 18, 20, 25, 239, 145, 99

	0	1	2	3	4	5	6	7	8	9
006	6									
018	18									
020	20									
025	25									
234			234							
145		145								
099	99									

arr ⇒ 6, 18, 20, 25, 99, 145, 234

The arr is sorted

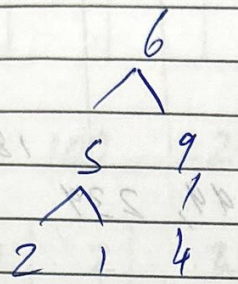
Q6)

6, 5, 9, 2, 1, 7

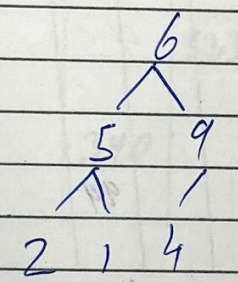
$n = 6$

$(n/2) - 1 = 3 - 1$

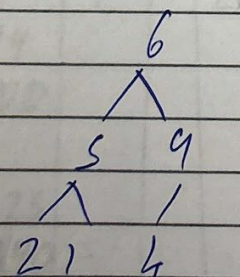
$= 2 \rightarrow$ index



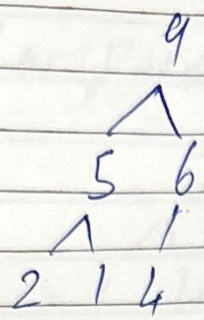
Compare 9 with 4



Compare 5 with 2 and 1



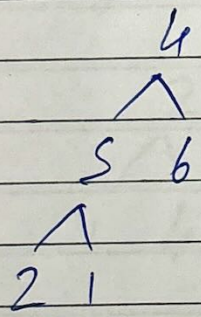
Compare 6 with 5 and 9



9, 5, 6, 2, 1, 4

Swap 9 with 4

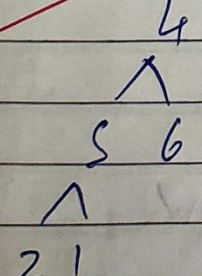
4, 5, 6, 2, 1, [9]



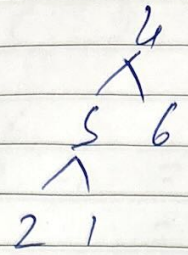
n = 5

(n/2) - 1

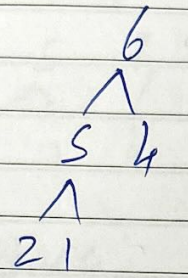
= 1 → index



Compare 5 with 2 and 1



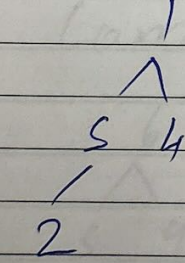
Compare 4 with 5 and 6



6, 5, 4, 2, 1, 9

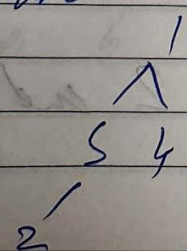
swap 1 and 6

1, 5, 4, 2, 6, 9

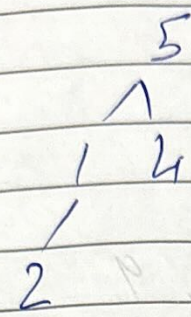


(n/2) - 1 = (4/2) - 1 = 2 - 1 = 1

Compare 5 with 2



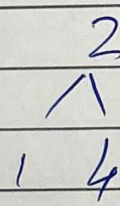
Compare 1 with 5 and 4



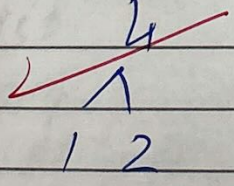
5, 1, 4, 2, 6, 9

Swap 5 with 2

2, 1, 4, ~~4~~ 5, 6, 9



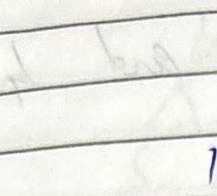
Compare 2 with 1 & 4



4, 1, 2, 5, 6, 9

Swap 2 & 4

2, 1, 4, ~~4~~ 5, 6, 9



Compare 2 and 1

2, 1, 4, 5, 6, 9

Swap 1 & 2

Sorted array:

1, 2, 4, 5, 6, 9

4/9/25

Extra

Algorithms for all sorting techniques (7)

1. Bubble Sort

Step:

1. Compare current number (j) with next number element [j+1] and swap/arrange such that a[j] < a[j+1] until a[n] / the last element is the largest element
2. Exclude last sorted element and repeat step 1 such that next largest (i.e second or third largest) element is found
3. Repeat ~~next~~ step 2 until sorted list is produced (at maximum n-1 repetitions of step 2 will occur)

2. Selection Sort

Steps:

1. Start from the first element in the list and compare it with each individual element in the list such that a[] < each element. Swap in comparisons if necessary to satisfy the condition.
2. Exclude the smallest element and repeat Step 1 such that the next smallest element is found.
3. Repeat step 2 for each position / element until only one element remains unsorted (the largest)

So, sorted list is produced.

3. Insertion Sort.

Steps:

1. If it is the first element it is already sorted. Return 1.
2. Pick the next element.
3. Compare with all elements in sorted sub-list.
4. Shift all elements in the sorted sub-list that are greater than the value to be sorted by one place.
5. Insert the value.
6. Repeat until list is sorted.

4. Radix Sort

Steps:

1. Define 10 queues, each representing a bucket for each digit from 0 to 9
2. Consider least significant digit of each number in the list to be sorted.
3. Insert each number into their respective bucket based on value of least significant digit
4. Group all the numbers from quere 0 to quere 9, in order they have been inserted into their respective queues.
5. Repeat from step 3 based on next least significant digit.
6. Repeat from step 2 until all the numbers are grouped based on most significant digit

5. Shell Sort

Steps:

1. Start
2. Initialize
3. Divide list equal in
4. Modify gap sequence
2. Sort sub
5. Modify sequency
6. Repeat
7. Print
8. Stop

6. Heap

Steps:

1. Construct of si
2. Find and heap heap
3. Recons element unsorte
2. Recons largest
5. Repeat is
6. Print