## Experiment No 4.1

Date:

**AIM**: Implementation of sorting algorithms -

1. Bubble
2. Insertion
3. Selection
4. Quick
5. Merge
6. Heap

## Experiment No 4.1.1

Date:

## BUBBLE SORT

**AIM:** Write a program to implement bubble sort

### ALGORITHM

**PROGRAM**

```
#include<stdio.h>
int size;

void printList(int ar[]){
    for (int i = 0; i < size; i++){
           printf("%d\t",ar[i]);
    }
    printf("\n");
}

void bubbleSort(int ar[]){
    for(int i = 0;i < size-1;i++){
           for(int j=0;j<size-i-1;j++ ){
                     if (ar[j] > ar[j + 1]){
                              int temp = ar[j];
                              ar[j] = ar[j + 1];
                              ar[j + 1] = temp;
                     }
           }
           printf("Iteration %d\n",i);
           printList(ar);
    }
}

int main(){
    printf("Enter no.of elements : ");
    scanf("%d",&size);
    int ar[size];
    printf("Enter the elements\n");
    for (int i = 0; i < size; i++){
           scanf("%d",&ar[i]);
    }
    printf("List before sorting\n");
    printList(ar);
    bubbleSort(ar);
    printf("List after sorting\n");
    printList(ar);
    return 0;
}
```

**OUTPUT**

```
Enter no.of elements : 5
Enter the elements
4
6
8
```

```
4
1
List before sorting
4       6       8       4       1
Iteration 0
4       6       4       1       8
Iteration 1
4       4       1       6       8
Iteration 2
4       1       4       6       8
Iteration 3
1       4       4       6       8
List after sorting
1       4       4       6       8
```

## Experiment No 4.1.2

Date:

## INSERTION SORT

**AIM:**Write a program to implement insertion sort

### ALGORITHM

**PROGRAM**

```
#include <stdio.h>
void insertionsort(int arr[],int n)
{
  printf("Sorting using Insertion Sort :\n");
  for (int i=1;i<n;i++)
  {
          int key=arr[i];
          int j=i-1;
          while (key<arr[j] && j>=0)
          {
          arr[j+1]=arr[j];
          j--;
          }
          arr[j+1]=key;
          for (int i=0;i<n;i++)
          {
          printf("%d ", arr[i]);
          }
          printf("\n");
  }
}


void main()
{
          int n,arr[100];
            printf("Enter number of elements :");
          scanf("%d", &n);
          printf("Enter %d integers for sorting :\n", n);

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


}
```

## **OUTPUT**

```
Enter number of elements :6
Enter 6 integers for sorting :
9 1 7 2 4 3
Sorting using Insertion Sort :
1 9 7 2 4 3
1 7 9 2 4 3
1 2 7 9 4 3
1 2 4 7 9 3
1 2 3 4 7 9
Sorted array is :
1 2 3 4 7 9
```

## Experiment No 4.1.3

Date:

## SELECTION SORT

**AIM**:Write a program to implement selection sort

### ALGORITHM

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int size, arr[50], i, j, temp, small, count=0, index;
        printf("Enter size for Array: ");
        scanf("%d", &size);
        printf("Enter %d array elements: ", size);
        for(i=0; i<size; i++)
        scanf("%d", &arr[i]);
        for(i=0; i<(size-1); i++)
        {
        small = arr[i];
        for(j=(i+1); j<size; j++)
        {
        if(small>arr[j])
        {
                small = arr[j];
                count++;
                index = j;
        }
        }
        if(count!=0)
        {
        temp = arr[i];
        arr[i] = small;
        arr[index] = temp;
        }
        printf("\nStep %d: ", i+1);
        for(j=0; j<size; j++)
        printf("%d ", arr[j]);
        printf("\n");
        count=0;
}
        printf("\nNow the Array after sorting is:\n");
        for(i=0; i<size; i++)
        printf("%d ", arr[i]);
        getch();
        return 0;
}
```

**OUTPUT**

```
Enter size for Array: 5
Enter 5 array elements: 5
```

```
2
8
9
4

Step 1: 2 5 8 9 4

Step 2: 2 4 8 9 5

Step 3: 2 4 5 9 8

Step 4: 2 4 5 8 9

Now the Array after sorting is:
2 4 5 8 9
```

## Experiment No 4.1.4

Date:

### QUICK SORT

**AIM**: Write a program to implement quick sort

### ALGORITHM

**PROGRAM**

```
#include <stdio.h>
int n;
// function to swap elements
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
// function to print array elements
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}
// function to find the partition position
int partition(int array[], int low, int high) {

  // select the rightmost element as pivot
  int pivot = array[high];

  // pointer for greater element
  int i = (low - 1);

  // traverse each element of the array
  // compare them with the pivot
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {

      // if element smaller than pivot is found
      // swap it with the greater element pointed by i
      i++;

      // swap element at i with element at j
      swap(&array[i], &array[j]);
    }
  }

  // swap the pivot element with the greater element at i
  swap(&array[i + 1], &array[high]);

  // return the partition point
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {

    // find the pivot element such that
    // elements smaller than pivot are on left of pivot
    // elements greater than pivot are on right of pivot
    int pi = partition(array, low, high);
    printArray(array,n);
    // recursive call on the left of pivot

    quickSort(array, low, pi - 1);

    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
  }
}

// main function
int main() {
  int i;
  printf("enter size of array:");
  scanf("%d",&n);
  int data[n];
  printf("enter elements:");
  for(i=0;i<n;i++){
      scanf("%d",&data[i]);
  }


  printf("Unsorted Array\n");
  printArray(data, n);
  printf("Intermediate array\n");

  // perform quicksort on data
  quickSort(data, 0, n - 1);

  printf("Sorted array in ascending order: \n");
  printArray(data, n);
}
```

**OUTPUT**

```
enter size of array:5
enter elements:3
8
2
5
1
Unsorted Array
3  8  2  5  1
```

```
Intermediate array
1  8  2  5  3
1  2  3  5  8
1  2  3  5  8
Sorted array in ascending order:
1  2  3  5  8
```

## Experiment No 4.1.5

Date:

### MERGE SORT

**AIM**: Write a program to implement merge sort

### ALGORITHM

**PROGRAM**

```
#include <stdio.h>

void printArray(int A[], int n)
{
        for (int i = 0; i < n; i++)
        {
        printf("%d ", A[i]);
        }
        printf("\n");
}

void merge(int A[], int mid, int low, int high)
{
        int i, j, k, B[100];
        i = low;
        j = mid + 1;
        k = low;

        while (i <= mid && j <= high)
        {
        if (A[i] < A[j])
        {
        B[k] = A[i];
        i++;
        k++;
        }
        else
        {
        B[k] = A[j];
        j++;
        k++;
        }
        }
        while (i <= mid)
        {
        B[k] = A[I];
k++;
        i++;
        }
        while (j <= high)
        {
        B[k] = A[j];
        k++;
        j++;
        }
        for (int i = low; i <= high; i++)
        {
```

```
        A[i] = B[i];
        }

}

void mergeSort(int A[], int low, int high){
        int mid;
        if(low<high){
        mid = (low + high) /2;
        mergeSort(A, low, mid);
        mergeSort(A, mid+1, high);
        printArray(A, high+1);
        merge(A, mid, low, high);
        }
}

int main()
{
        int n,A[100];
        printf("enter the number of elements of the array ");
        scanf("%d",&n);
        printf("Enter the elements ");
          for(int i=0;i<n;i++)
        {
        scanf("%d",&A[i]);
        }
        printf("the input array is \n");
        printArray(A, n);
        printf("\n");
        mergeSort(A, 0, n-1);
        printf("\n the sorted array is \n");
        printArray(A, n);
        return 0;
}
```

**OUTPUT**

```
enter the number of elements of the array 7
Enter the elements 9 1 4 14 3 6 5
the input array is
9 1 4 14 3 6 5

9 1
1 9 4 14
1 9 4 14
1 4 9 14 3 6
1 4 9 14 3 6 5
1 4 9 14 3 5 6
```

```
 the sorted array is
1 3 4 5 6 9 14
```

## Experiment No 1.5

Date:

### HEAP SORT

**AIM**: Write a program to implement heap sort

### ALGORITHM

## PROGRAM

```
#include <stdio.h>
void swap(int*,int*);
void heapify(int arr[],int,int);
void heapsort(int arr[],int);
void print_array(int arr[],int);
void swap(int *p,int *q){

  int temp=*p;
  *p=*q;
  *q=temp;}


void heapify(int arr[],int n,int i){

        int largest =i;
        int left =2*i+1;
        int right =2*i+2;
        if (left <n && arr[left]>arr [largest]){
                largest=left;}
        if (right <n && arr[right]>arr [largest]){
                largest=right;}

        if (largest!=i){
        swap(&arr[i],&arr[largest]);
        heapify(arr,n,largest);}}

void heapsort(int arr[],int n){
        for(int i=n/2-1;i>=0;i--)
        {heapify(arr, n, i);
        }

        for (int i=n-1;i>=0;i--){
        swap(&arr[0], &arr[i]);
        heapify(arr,i,0);
        print_array(arr,n);
        }}
void print_array(int arr[], int n)
{
        for (int m = 0; m<n ;m++)
        printf("%d ", arr[m]);
        printf("\n");
}

void main(){
  int n,i;
  printf("Enter no of elements:");
  scanf("%d",&n);
  int arr[n];
  printf("\nEnter array elements:\n");
  for(int j=0;j<n;j++){
        scanf("%d",&arr[j]);}
  printf("\nSorting using heap sort \n");
  heapsort(arr, n);

  printf("\nSorted array is \n");
  print_array(arr, n);


        }
```

**OUTPUT**

```
Enter no of elements:7

Enter array elements:
45
8
0
12
3
5
16

Sorting using heap sort
16 12 5 8 3 0 45
12 8 5 0 3 16 45
8 3 5 0 12 16 45
5 3 0 8 12 16 45
3 0 5 8 12 16 45
0 3 5 8 12 16 45
0 3 5 8 12 16 45

Sorted array is
0 3 5 8 12 16 45
```

## Experiment No 4.2

Date:

**AIM**: Implementation of searching algorithms

1. Linear search
2. Binary search

## Experiment No 4.2.1

Date:

### LINEAR SEARCH

**AIM**: Write a program to implement Linear Search

### ALGORITHM

**PROGRAM**

```
#include <stdio.h>

int main()
{
  int array[100], search, c, n, count = 0;

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

  printf("Enter %d numbers\n", n);

  for (c = 0; c < n; c++)
        scanf("%d", &array[c]);

  printf("Enter a number to search\n");
  scanf("%d", &search);

  for (c = 0; c < n; c++) {
        if (array[c] == search) {
        printf("%d is present at location %d.\n", search, c+1);
        count++;
        }
  }
  if (count == 0)
        printf("%d isn't present in the array.\n", search);
  else
        printf("%d is present %d times in the array.\n", search, count);

  return 0;
}
```

**OUTPUT**

```
Enter number of elements in array
5
Enter 5 numbers
2
9
4
8
3
Enter a number to search
4
4 is present at location 3.
4 is present 1 times in the array.
```

## Experiment No 4.2.1

Date:

### BINARY SEARCH

**AIM**: Write a program to implement Binary Search

### ALGORITHM

**PROGRAM**

```
#include  <stdio.h>
int readArray(int a[],int n)
{
	printf("Enter the elements: ");
	int i;
	for(i=0;i<n;i++)	{
	scanf("%d",a+i);
	}
}
int binarySearch(int a[],int first,int last,int key)
{
	int mid,flag=0;
	while(first<=last)
	{
	mid = (first+last)/2;
	if(a[mid]==key)
	{
	printf("element found at pos %d",mid);
	}
	else if(a[mid]>key)
	{
	binarySearch(a,first,mid-1,key);
	}
	else
	{
	binarySearch(a,mid+1,last,key);
	}
	flag=1;
	break;
	}
	if(!flag)
	{
	printf("element not found");
	}
}
int main()
{
	int n,key;
	system("clear");
	printf("enter the number of elements: ");
	scanf("%d",&n);
	int a[n];
	readArray(a,n);
	printf("enter the key: ");
	scanf("%d",&key);
	binarySearch(a,0,n-1,key);
}
```

**OUTPUT**

```
enter the number of elements: 7
Enter the elements: 8 13 20 78 100 122 154
enter the key: 100
element found at pos 4
```