**EXP 1: Implement insertion sort**

#include <math.h>

#include <stdio.h>

void insertionSort(int arr[], int n)

{

int i, key, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

{

int i;

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

printf("%d ", arr[i]);

printf("\n");

}

int main()

{

int arr[] = { 15, 12, 20, 7, 10 };

int n = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

Output:-

7,10,12,15,20

**EXP 2: Implement selection sortr**

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

for (i = 0; i < n-1; i++)

{

min\_idx = i;

for (j = i+1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

if(min\_idx != i)

swap(&arr[min\_idx], &arr[i]);

}

}

void printArray(int arr[], int size)

{

int i;

for (i=0; i < size; i++)

printf("%d ", arr[i]);

printf("\n");

}

int main()

{

int arr[] = {64, 25, 12, 22, 11};

int n = sizeof(arr)/sizeof(arr[0]);

selectionSort(arr, n);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}

**OUTPUT:**



**EXP 3: Implement Binary sort**

#include <stdio.h>

int binarySearch(int a[], int item, int low, int high)

{

if (high <= low)

return (item > a[low]) ? (low + 1) : low;

int mid = (low + high) / 2;

if (item == a[mid])

return mid + 1;

if (item > a[mid])

return binarySearch(a, item, mid + 1, high);

return binarySearch(a, item, low, mid - 1);

}

void insertionSort(int a[], int n)

{

int i, loc, j, k, selected;

for (i = 1; i < n; ++i) {

j = i - 1;

selected = a[i];

loc = binarySearch(a, selected, 0, j);

while (j >= loc) {

a[j + 1] = a[j];

j--;

}

a[j + 1] = selected;

}

}

int main()

{

int a[]

= { 37, 23, 0, 17, 12, 72, 31, 46, 100, 88, 54 };

int n = sizeof(a) / sizeof(a[0]), i;

insertionSort(a, n);

printf("Sorted array: \n");

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

printf("%d ", a[i]);

return 0;

}

**OUTPUT:**

