#include <stdio.h>

void swap (int \*v1 , int \*v2){

    int temp = \*v1;

    \*v1 = \*v2;

    \*v2= temp;

};

void bubbleSort(int arr[],int lenght){

     for(int i=0 ; i <lenght;i++){

        for(int j=0 ; j<lenght-i-1;j++){

            if(arr[j]>arr[j+1]){

                swap(&arr[j],&arr[j+1]);

            }

        }

    }

};

void selectionSort(int arr[],int length){

    for(int i = 0; i<length;i++){

        int minInx = i;

        for (int j=i+1;j<length;j++){

            if(arr[minInx]>arr[j]){

                minInx= j;

            }

        }

        swap(&arr[minInx],&arr[i]);

    }

};

void insertionSort(int arr[], int length) {

    for (int i = 1; i < length; i++) {

        int temp = arr[i];

        int j = i - 1;

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

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = temp;

    }

}

void bucketSort(int arr[], int length){

    int bucket[length][length];

    int bucketCount[length];

    for(int i = 0 ; i<length;i++){

        bucketCount[i]=0;

    }

    int bi = 0;

    for(int i= 0;i<length;i++){

        bi=length\*arr[i];

        bucket[bi][bucketCount[bi]++]=arr[i];

    }

    for(int i = 0 ; i <length;i++){

        insertionSort(bucket[i],bucketCount[i]);

    }

    int index = 0;

    for (int i = 0 ; i <length ; i++){

        for (int j = 0 ; j<bucketCount[i];j++){

            arr[index++] = bucket[i][j];

        }

    }

}

int partition(int arr[], int low, int high) {

    int pivot = arr[high];

    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {

        if (arr[j] < pivot) {

            i++;

            swap(&arr[i], &arr[j]);

        }

    }

    swap(&arr[i + 1], &arr[high]);

    return i + 1;

}

void quickSort(int arr[], int low, int high) {

    if (low < high) {

        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

void print(int arr[],int lenght){

    for(int i = 0 ;i<lenght;i++){

        printf("%d,",arr[i]);

    }

}

int main (){

    int arr1[] = {4232,423,4,5,57,89,6,565,5,9,7,0};

    float arr2[]= {0.89,0.67,0.89,0.65,0.21};

    int len1 = sizeof(arr1)/sizeof(int);

    int len2 = sizeof(arr2)/sizeof(float);

    int option;

    printf("Enter the option:" );

    scanf("%d",&option);

         switch (option)

        {

        case 1:

            printf("Bubble Sort\n");

            printf("before Sort -> ");

            print(arr1,len1);

            bubbleSort(arr1,len1);

            printf("\nafter Sort -> ");

            print(arr1,len1);

            break;

        case 2:

            printf("Selection Sort\n");

            printf("before Sort -> ");

            print(arr1,len1);

            selectionSort(arr1,len1);

            printf("\nafter Sort ->  ");

            print(arr1,len1);

            break;

        case 3:

            printf("insersion Sort\n");

            printf("before Sort -> ");

            print(arr1,len1);

            insertionSort(arr1,len1);

            printf("\nafter Sort ->  ");

            print(arr1,len1);

            break;

        case 4:

            printf("Bucket Sort\n");

            printf("before Sort -> ");

            print(arr1,len1);

            bucketSort(arr1,len1);

            printf("\nafter Sort ->  ");

            print(arr1,len1);

            break;

        case 5:

            printf("Quick Sort\n");

            printf("before Sort -> ");

            print(arr1,len1);

            quickSort(arr1,0,len1-1);

            printf("\nafter Sort ->  ");

            print(arr1,len1);

            break;

        default:

            break;

        }

        return 0;

}

FIBONACCHI SEARCH

#include <stdio.h>

int fibonacciSearch(int arr[], int n, int x) {

    int fibMMm2 = 0;

    int fibMMm1 = 1;

    int fibM = fibMMm2 + fibMMm1;

    while (fibM < n) {

        fibMMm2 = fibMMm1;

        fibMMm1 = fibM;

        fibM = fibMMm2 + fibMMm1;

    }

    int offset = -1;

    while (fibM > 1) {

        int i = (offset + fibMMm2 < n - 1) ? offset + fibMMm2 : n - 1;

        if (arr[i] < x) {

            fibM = fibMMm1;

            fibMMm1 = fibMMm2;

            fibMMm2 = fibM - fibMMm1;

            offset = i;

        } else if (arr[i] > x) {

            fibM = fibMMm2;

            fibMMm1 = fibMMm1 - fibMMm2;

            fibMMm2 = fibM - fibMMm1;

        } else {

            return i;

        }

    }