#include<iostream>

using namespace std;

void merge(int *a*[], int *l*, int *m*, int *r*)

{

    int n1 = *m* - *l* + 1;

    int n2 = *r* - *m*;

    int\* L = **new** int[n1];

    int\* R = **new** int[n2];

    for (int i = 0; i < n1; i++)

        L[i] = *a*[*l* + i];

    for (int j = 0; j < n2; j++)

        R[j] = *a*[*m* + 1 + j];

    int i = 0;

    int j = 0;

    int k = *l*;

    while (i < n1 && j < n2)

    {

        if (L[i] <= R[j])

        {

*a*[k] = L[i];

            i++;

        }

        else

        {

*a*[k] = R[j];

            j++;

        }

        k++;

    }

    while (i < n1)

    {

*a*[k] = L[i];

        i++;

        k++;

    }

    while (j < n2)

    {

*a*[k] = R[j];

        j++;

        k++;

    }

}

void mergeSort(int *a*[], int *l*, int *r*)

{

    if (*l* < *r*)

    {

        int m = *l* + (*r* - *l*) / 2;

        mergeSort(*a*, *l*, m);

        mergeSort(*a*, m + 1, *r*);

        merge(*a*, *l*, m, *r*);

    }

}

void shakerSort(int *a*[], int *n*) {

    int left, right, k, i;

    left = 0;

    right = *n* - 1;

    k = *n* - 1;

    while (left < right) {

        for (i = right; i > left; i--)

            if (*a*[i] < *a*[i - 1]) {

                int tg = *a*[i];

*a*[i] = *a*[i - 1];

*a*[i - 1] = tg;

                k = i;

            }

        left = k;

        for (int j = left; j < right; j++)

            if (*a*[j] > *a*[j + 1]) {

                int tg = *a*[j];

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

*a*[j + 1] = tg;

                k = j;

            }

        right = k;

    }

}

void bumbleSort(int *a*[], int *n*) {

    for(int i = 0; i < *n* - 1; i++){

        for(int j = *n* - 1; j < i ;j--){

            if(*a*[j] < *a*[j - 1]){

                int tg = *a*[j];

*a*[j] = *a*[j - 1];

*a*[j - 1] = tg;

            }

        }

    }

}

void selectionSort(int *a*[],int *n*){

    int temp;

    for(int i = 0; i < *n* - 1;i++){

        int min = 1;

        for (int j = i + 1; j < *n*; j++){

            if(*a*[min] > *a*[i])

            min = j;

        }

        if(min != i )

        temp = *a*[i];

*a*[i] = *a*[min];

*a*[min] = temp;

    }

}

void insertionSort(int *a*[], int *n*)  {

    for (int i = 1; i < *n*; i++)

    {

        int key = *a*[i];

        int pos = i - 1;

        while (pos >= 0 && *a*[pos] > key)

        {

*a*[pos + 1] = *a*[pos];

            --pos;

        }

*a*[pos + 1] = key;

    }

}

void quickSort(int *a*[], int *l*, int *r*) {

    int tmp;

    int i = *l*;

    int j = *r*;

    int x = *a*[(*l* + *r*) / 2];

    do {

        while (*a*[i] < x) i++;

        while (*a*[j] > x) j--;

        if (i <= j) {

            tmp = *a*[j]; *a*[j] = *a*[i]; *a*[i] = tmp;

            i++; j--;

        }

    } while (i <= j);

    if (*l* < j) quickSort(*a*, *l*, j);

    if (*r* > i) quickSort(*a*, i, *r*);

}

void nhap(int *a*[], int *n*) {

    cout << "Nhập các phần tử mảng: " << endl;

    for (int i = 0; i < *n*; i++) {

        cout << "a[" << i << "] = ";

        cin >> *a*[i];

    }

}

void xuat (int *a*[], int *n*){

    int i;

    for (i = 0; i < *n*; i++)

    cout << *a*[i]<<" ";

    cout << endl ;

}

int main(){

    int arr[] = {100};

    int n = sizeof(arr);

    cout << "Nhập số lượng phần tử của mảng: ";

    nhap;

    cout <<"-------------------------------------------";

    cout <<"Chọn thuật toán sắp xếp Merge Sort nhấn phím 1";

    cout <<"Chọn thuật toán sắp xếp Shaker Sort nhấn phím 2";

    cout <<"Chọn thuật toán sắp xếp Bumble Sort nhấn phím 3";

    cout <<"Chọn thuật toán sắp xếp Selection Sort nhấn phím 4";

    cout <<"Chọn thuật toán sắp xếp Insertion Sort nhấn phím 5";

    cout <<"Chọn thoát chương trình nhấn phím 0";

    int option = 0;

    switch (option) {

        case 1:

        mergeSort(arr, 0, n - 1);

        break;

        case 2:

        shakerSort(arr, n);

        break;

        case 3:

        bumbleSort(arr, n);

        break;

        case 4:

        selectionSort(arr, n);

        break;

        case 5:

        insertionSort(arr, n);

        break;

        case 6:

        quickSort;

        break;

        defalt:

        cout << "Thoát chương trình";

    }

    xuat;

    return 0;

}