**CODE:-**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

int LinearSearch(int arr[], int n, int item)

{

int i = 0;

while (i < n && item > arr[i])

i++;

if (arr[i] == item)

return i;

else

return -1;

}

int BinarySearch(int arr[], int low, int up, int item)

{

int mid;

if (low > up)

return -1;

mid = (low + up) / 2;

if (item > arr[mid])

BinarySearch(arr, mid + 1, up, item);

else if (item < arr[mid])

BinarySearch(arr, low, mid - 1, item);

else

return mid;

}

void BubbleSort(int arr[], int n)

{

int i, j, temp, exchange;

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

{

exchange = 0;

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

{

if (arr[j] > arr[j + 1])

{

temp = arr[j];

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

arr[j + 1] = temp;

exchange++;

}

}

if (!exchange)

break;

}

}

void SelectionSort(int arr[], int n)

{

int i, j, min, temp;

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

{

min = i;

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

{

if (arr[min] > arr[j])

min = j;

}

if (i != min)

{

temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

}

}

void InsertionSort(int arr[], int n)

{

int i, j, k;

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

{

k = arr[i];

for (j = i - 1; j >= 0 && k < arr[j]; j--)

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

arr[j + 1] = k;

}

}

void merge(int arr[], int temp[], int low1, int up1, int low2, int up2)

{

int i = low1;

int j = low2;

int k = low1;

while ((i <= up1) && (j <= up2))

{

if (arr[i] <= arr[j])

temp[k++] = arr[i++];

else

temp[k++] = arr[j++];

}

while (i <= up1)

temp[k++] = arr[i++];

while (j <= up2)

temp[k++] = arr[j++];

}

void copy(int arr[], int temp[], int low, int up)

{

int i;

for (i = low; i <= up; i++)

arr[i] = temp[i];

}

void MergeSort(int arr[], int low, int up)

{

int mid;

int temp[MAX];

if (low < up)

{

mid = (low + up) / 2;

MergeSort(arr, low, mid);

MergeSort(arr, mid + 1, up);

merge(arr, temp, low, mid, mid + 1, up);

copy(arr, temp, low, up);

}

}

void printArray(int arr[], int n)

{

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

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

}

int main()

{

int ch, size, elem, index;

while (1)

{

printf("Enter the size of the array.Enter -1 to exit.\n");

scanf("%d", &size);

if (size == -1)

exit(1);

else

{

int arr[size];

printf("Enter the array.\n");

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

scanf("%d", &arr[i]);

printf("Enter 1 to do linear search.\n");

printf("Enter 2 to do binary search.\n");

printf("Enter 3 to do bubble sort.\n");

printf("Enter 4 to do selection sort.\n");

printf("Enter 5 to do insertion sort.\n");

printf("Enter 6 to do merge sort.\n");

printf("Enter 7 to exit.\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter element to be searched.\n");

scanf("%d", &elem);

index = LinearSearch(arr, size, elem);

if (index == -1)

printf("Element not found.\n");

else

printf("Element found at index %d in array.\n", index);

break;

case 2:

printf("Enter element to be searched.\n");

scanf("%d", &elem);

BubbleSort(arr, size);

index = BinarySearch(arr, 0, size - 1, elem);

if (index == -1)

printf("Element not found.\n");

else

printf("Element found at index %d in sorted array.\n", index);

break;

case 3:

printf("Sorted array is:\n");

BubbleSort(arr, size);

printArray(arr, size);

printf("\n");

break;

case 4:

printf("Sorted array is:\n");

SelectionSort(arr, size);

printArray(arr, size);

printf("\n");

break;

case 5:

printf("Sorted array is:\n");

InsertionSort(arr, size);

printArray(arr, size);

printf("\n");

break;

case 6:

printf("Sorted array is:\n");

MergeSort(arr, 0, size - 1);

printArray(arr, size);

printf("\n");

break;

case 7:

exit(1);

default:

printf("Erroneous input.\n");

}

}

}

}

**OUTPUT:-**







