#include <stdio.h>

int linear\_search(int arr[], int n, int val){

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

if (val == arr[i])

return i;

}

return -1;

}

int main(){

int n = 0;

printf("No. of array elements : ");

scanf("%d", &n);

int arr[n];

printf("\nEnter array elements : ");

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

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

}

printf("Value to search in array : ");

int val;

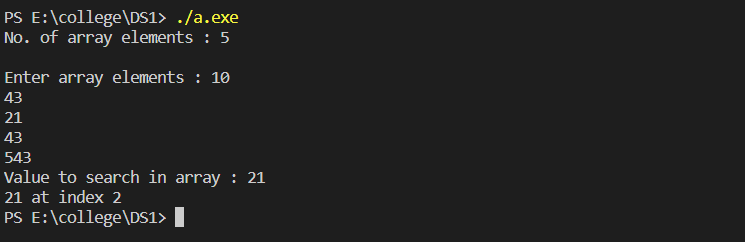
scanf("%d", &val);

int res = linear\_search(arr, n, val);

printf("%d at index %d", val, res);

return 0;

}



#include <stdio.h>

int binarysearch(int arr[], int val, int start, int end){

    int mid;

    if(start <= end){

        mid = (start + end)/2;

    }

    if(arr[mid] == val){

        return mid+1;

    }

    else if(val > arr[mid]){

        return binarysearch(arr, val, mid+1, end);

    }

    else {

        return binarysearch(arr, val, start, mid-1);

    }

    return -1;

}

int main(){

    int n = 0;

    printf("No. of array elements : ");

    scanf("%d", &n);

    int arr[n];

    printf("\nEnter array elements in ascending order : ");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Value to search in array : ");

    int val;

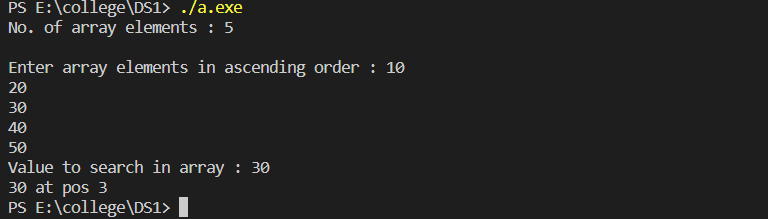
    scanf("%d", &val);

    int res = binarysearch(arr, val, 0, n-1);

    printf("%d at pos %d", val, res);

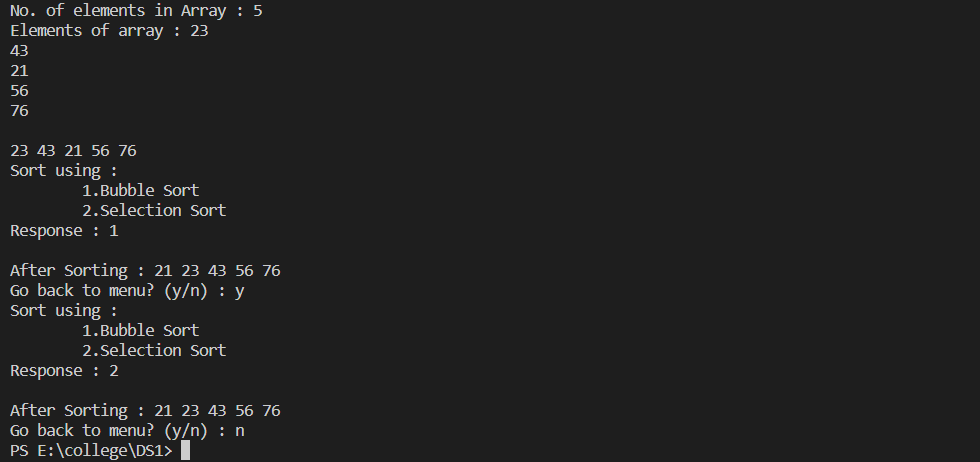
    return 0;

}



#include <stdio.h> //prac5  
void swap(int \*a, int \*b){  
    int \*temp = a;  
    a = b;  
    b = temp;  
}  
void bubblesort(int arr[], int n){  
    int temp;  
    for (int i = 0; i < n; i++) {  
        for(int j = 0; j < n-1; j++){  
            if (arr[j] > arr[j+1]){  
                temp = arr[j];  
                arr[j] = arr[j+1];  
                arr[j+1] = temp;  
                // swap(&arr[j], &arr[j+1]);  
            }  
        }  
    }  
}  
  
void selectionsort(int arr[], int n){  
    int min, temp;  
    for(int i = 0; i < n-1; i++){  
        min = i;  
        for(int j=i+1; j < n; j++){  
            if(arr[j] < arr[min]){  
                min = j;  
            }  
        }  
        temp = arr[i];  
        arr[i] = arr[min];  
        arr[min] = temp;  
    }  
}  
  
int main()  
{  
    int response, n;  
    printf("No. of elements in Array : ");  
    scanf("%d", &n);  
     
    int arr[n];  
    char con;  
    printf("Elements of array : ");  
    for (int i = 0; i < n; i++) {  
        scanf("%d", &arr[i]);  
    }  
    printf("\n");  
    for (int i = 0; i < n; i++) {  
        printf("%d ", arr[i]);  
    }  
    printf("\n");  
     
    do{  
        printf("Sort using : \n\t1.Bubble Sort \n\t2.Selection Sort");  
        printf("\nResponse : ");  
        scanf("%d", &response);  
         
        switch(response){  
        case 1:  
        bubblesort(arr, n);  
        break;  
         
        case 2:  
        selectionsort(arr, n);  
        break;  
         
        default:  
        printf("\nSelect valid response");  
         
        }  
         
        printf("\nAfter Sorting : ");  
        for (int i = 0; i < n; i++) {  
            printf("%d ", arr[i]);  
        }  
         
        printf("\nGo back to menu? (y/n) : ");  
        scanf(" %c", &con);  
         
    } while(con == 'y');  
  
    return 0;

}



#include <stdio.h> //prac6

void swap(int \*a, int \*b) {

  int t = \*a;

  \*a = \*b;

  \*b = t;

}

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

  int pivot = a[high];

  int i = (low - 1);

  for (int j = low; j < high; j++) {

    if (a[j] <= pivot) {

      i++;

      swap(&a[i], &a[j]);

    }

  }

  swap(&a[i + 1], &a[high]);

  return (i + 1);

}

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

  if (low < high) {

    int pa = partition(a, low, high);

    quickSort(a, low, pa - 1);

    quickSort(a, pa + 1, high);

  }

}

void merge(int arr[], int p, int q, int r) {

  int n1 = q - p + 1;

  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)

    L[i] = arr[p + i];

  for (int j = 0; j < n2; j++)

    M[j] = arr[q + 1 + j];

  int i, j, k;

  i = 0;

  j = 0;

  k = p;

  while (i < n1 && j < n2) {

    if (L[i] <= M[j]) {

      arr[k] = L[i];

      i++;

    } else {

      arr[k] = M[j];

      j++;

    }

    k++;

  }

  while (i < n1) {

    arr[k] = L[i];

    i++;

    k++;

  }

  while (j < n2) {

    arr[k] = M[j];

    j++;

    k++;

  }

}

void mergeSort(int arr[], int l, int r) {

  if (l < r) {

    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);

    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);

  }

}

int main(){

    int response, n;

    printf("No. of elements in Array : ");

    scanf("%d", &n);

    int arr[n];

    char con;

    do{

        printf("Elements of array : ");

        for (int i = 0; i < n; i++) {

            scanf("%d", &arr[i]);

        }

        printf("\n");

        for (int i = 0; i < n; i++) {

            printf("%d ", arr[i]);

        }

        printf("\n");

        printf("Sort using : \n\t1.Quick Sort \n\t2.Merge Sort");

        printf("\nResponse : ");

        scanf("%d", &response);

        switch (response){

            case 1:

            quickSort(arr, 0, n-1);

                break;

            case 2:

            mergeSort(arr, 0, n-1);

                break;

            default:

                printf("\nSelect valid response");

        }

        printf("\nAfter Sorting : ");

        for (int i = 0; i < n; i++) {

            printf("%d ", arr[i]);

        }

        printf("\nGo back to menu? (y/n) : ");

        scanf(" %c", &con);

    } while(con == 'y');

}

