****

**KRISHNA ENGINEERING COLLEGE**

**Data Structure Lab**

**KCS – 301**

SUBMITTED BY:-

**TALIB KHAN**

**1901610100214**

SUBMITTED TO:-

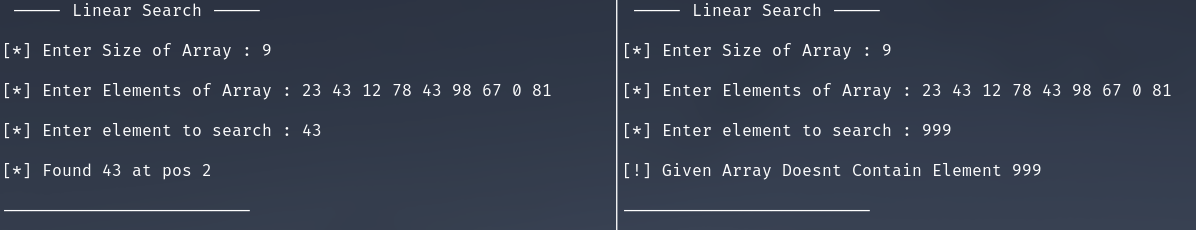
**Ms. RASHMI SHARMA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Krishna Engineering College**  **Department of Computer Science and Engineering** | | | |  |
| **Name: TALIB KHAN**  **Roll No.:1901610100214** | | |  |
|  |
| **Year/Semester:- 2nd/III** | | **Session:- 2020-21** |  |
| **Subject Name:- Data Structures** | | **Subject Code:- KCS-301** |  |
| **Faculty :-** | **Ms. RASHMI SHARMA** | **Section/ Group:**  **B/2** |  |
|  |  |  |  |
| **S.No** | **Experiment /Program Name** | **Scheduled Date** | **Sign** |
|
| 1 | Implementation of Linear search |  |  |
| 2 | Implementation of Binary search (using recursion and without using recursion) |  |  |
| 3 | Implementation of Selection sort |  |  |
| 4 | Implementation of Insertion sort |  |  |
| 5 | WAP in C to perform traversal of an array |  |  |

**Q1. WAP to implement Linear Search in C.**

|  |
| --- |
| #include <stdio.h> #define MAX 20  int linear(int array[], int n, int key);  void setup() {  printf("\n -----------------------");  printf("\n | Talib Khan |");  printf("\n | CSE 2B |");  printf("\n | Roll NO 1901610100214 |");  printf("\n ----------------------- ");  printf("\n\n"); }  int main() {  int i, n = MAX - 1, key, index;  int array[MAX];   printf(" ----- Linear Search -----\n\n");  printf("[\*] Enter Size of Array : ");   scanf("%d", &n);   printf("\n[\*] Enter Elements of Array : ");  for (i = 0; i < n; i++) {  scanf("%d", &array[i]);  }  printf("\n[\*] Enter element to search : ");  scanf("%d", &key);   index = linear(array, n, key);   if (index != -1) {  printf("\n[\*] Found %d at pos %d", key, index + 1);  } else {  printf("\n[!] Given Array Doesnt Contain Element %d ", key);  }   printf("\n\n-------------------------\n\n");  return 0; }  int linear(int array[], int n, int key) {  int i, idx = -1;  for (i = 0; i < n; i++) {  if (array[i] == key) {  idx = i;  break;  }  }  return idx; } |

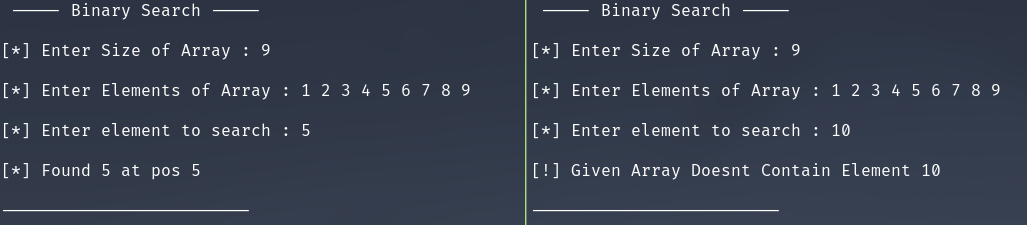
Output →



Q2(A). WAP to implement BINARY Search (Without Recursion) in C.

|  |
| --- |
| #include <stdio.h>  #define MAX 20  int binary(int array[], int n, int key);  void setup() {  printf("\n -----------------------");  printf("\n | Talib Khan |");  printf("\n | CSE 2B |");  printf("\n | Roll NO 1901610100214 |");  printf("\n ----------------------- ");  printf("\n\n"); }  int main() {  int i, n = MAX - 1, key, index;  int array[MAX];   printf(" ----- Binary Search -----\n\n");  printf("[\*] Enter Size of Array : ");   scanf("%d", &n);   printf("\n[\*] Enter Elements of Array : ");  for (i = 0; i < n; i++) {  scanf("%d", &array[i]);  }  printf("\n[\*] Enter element to search : ");  scanf("%d", &key);   index = binary(array, n, key);   if (index != -1) {  printf("\n[\*] Found %d at pos %d", key, index + 1);  } else {  printf("\n[!] Given Array Doesnt Contain Element %d ", key);  }   printf("\n\n-------------------------\n\n");  return 0; }  int binary(int array[], int n, int key) {  int idx, low, high, mid;   idx = -1;  low = 0;  high = n - 1;   while (low <= high) {  mid = low + (high - low) / 2; *// prevents overflow*  if (array[mid] == key) {  idx = mid;  break;  } else if (array[mid] < key) {  low = mid + 1;  } else { *// array[mid] > key*  high = mid - 1;  }  }  return idx; } |

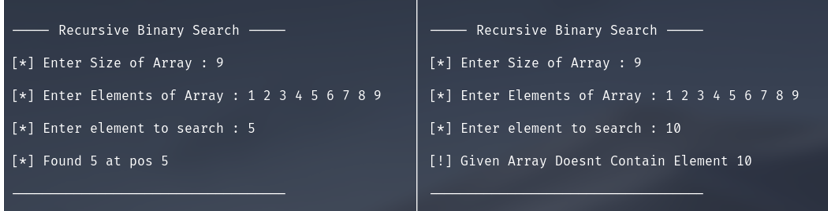
Output →



Q2(B). WAP to implement BINARY Search (Using Recursion) in C.

|  |
| --- |
| #include <stdio.h>  #define MAX 20  int binary(int array[], int low, int high, int key);  void setup() {  printf("\n -----------------------");  printf("\n | Talib Khan |");  printf("\n | CSE 2B |");  printf("\n | Roll NO 1901610100214 |");  printf("\n ----------------------- ");  printf("\n\n"); }  int main() {  int i, n = MAX - 1, key, index, low, high;  int array[MAX];  setup();  printf(" ----- Recursive Binary Search -----\n\n");  printf(" [\*] Enter Size of Array : ");   scanf("%d", &n);   printf("\n [\*] Enter Elements of Array : ");  for (i = 0; i < n; i++) {  scanf("%d", &array[i]);  }  printf("\n [\*] Enter element to search : ");  scanf("%d", &key);   low = 0;  high = n - 1;   index = binary(array, low, high, key);   if (index != -1) {  printf("\n [\*] Found %d at pos %d", key, index + 1);  } else {  printf("\n [!] Given Array Doesnt Contain Element %d ", key);  }   printf("\n\n -----------------------------------\n\n");  return 0; }  int binary(int array[], int low, int high, int key) {  int mid = low + (high - low) / 2; *// prevents overflow*   if (low > high) {  return -1;  } else if (array[mid] == key) {  return mid;  } else if (array[mid] < key) {  return binary(array, mid + 1, high, key);  } else { *// array[mid] > key*  return binary(array, low, mid - 1, key);  } } |

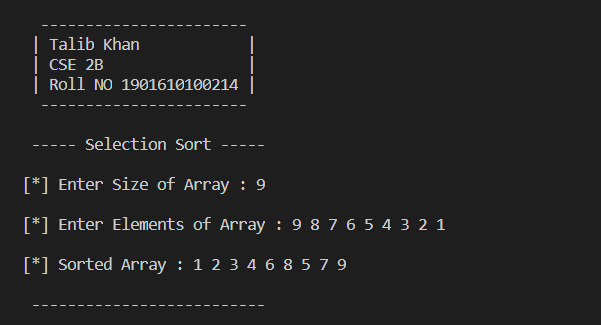
Output →



Q3. WAP to implement Selection Sorting in C.

|  |
| --- |
| #include <stdio.h>  #define MAX 10  void setup() {  printf("\n -----------------------");  printf("\n | Talib Khan |");  printf("\n | CSE 2B |");  printf("\n | Roll NO 1901610100214 |");  printf("\n ----------------------- ");  printf("\n\n"); } void selection\_sort(int \*array, int n);  int main() {  int i, n;  int array[MAX];  setup();  printf(" ----- Selection Sort -----\n\n");  printf("[\*] Enter Size of Array : ");   scanf("%d", &n);   printf("\n[\*] Enter Elements of Array : ");  for (i = 0; i < n; i++) {  scanf("%d", &array[i]);  }   selection\_sort(array, n);   printf("\n[\*] Sorted Array : ");   for (i = 0; i < n; i++) {  printf("%d ", array[i]);  }  printf("\n\n --------------------------\n\n");  return 0; }  void selection\_sort(int \*array, int n) {  int idx, temp;  for (int i = 0; i < n - 1; i++) {  for (int j = i; j < n; j++) {  if (array[i] > array[j]) idx = j;  }  temp = array[i];  array[i] = array[idx];  array[idx] = temp;  } } |

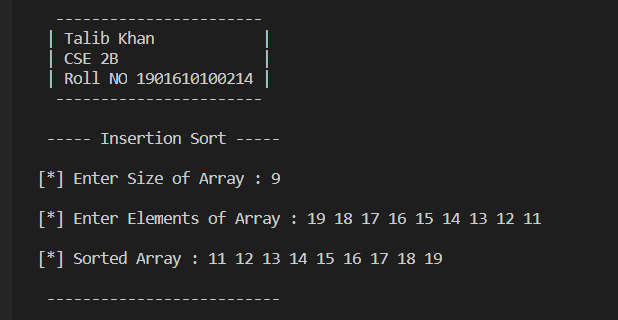
Output →



Q4. WAP to implement Insertion Sorting in C.

|  |
| --- |
| #include <stdio.h>  #define MAX 10  void insertion\_sort(int \*array, int n);  void setup() {  printf("\n -----------------------");  printf("\n | Talib Khan |");  printf("\n | CSE 2B |");  printf("\n | Roll NO 1901610100214 |");  printf("\n ----------------------- ");  printf("\n\n"); }  int main() {  int i, n;  int array[MAX];  setup();  printf(" ----- Insertion Sort -----\n\n");  printf("[\*] Enter Size of Array : ");   scanf("%d", &n);   printf("\n[\*] Enter Elements of Array : ");  for (i = 0; i < n; i++) {  scanf("%d", &array[i]);  }   insertion\_sort(array, n);   printf("\n[\*] Sorted Array : ");   for (i = 0; i < n; i++) {  printf("%d ", array[i]);  }  printf("\n\n --------------------------\n\n");  return 0; }  void insertion\_sort(int \*array, int n) {  int temp\_idx, temp;  for (int i = 1; i < n; i++) {  temp\_idx = i - 1;  temp = array[i];  while (temp\_idx >= 0 && array[temp\_idx] > temp) {  array[temp\_idx + 1] = array[temp\_idx];  temp\_idx--;  }  array[temp\_idx + 1] = temp;  } } |

Output →



Q5. WAP in C to perform traversal of an array.

#include<stdio.h>

#include<conio.h>

Void main()

{

Int n, i ,a[i];

Printf(“enter the limit of array\n”);

Scanf(“%d”,&n);

Printf(“enter an array\n”);

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

scanf(“%d”,&a[i]);

}

Printf(“ your array \n”);

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

printf(“%d”,&a[i]);

}

getch();

}

Output →

