   int result = binarySearchRecursive(charArray, 0, n - 1, target);

    if (result != -1) {

        printf("Character '%c' found at index %d.\n", target, result);

    } else {

        printf("Character '%c' not found in the array.\n", target);

    }

    return 0;

OUTPUT  
  
Enter the character to search (e to exit): R

low mid high

Iteration: 0 4 9

Iteration: 5 7 9

Iteration: 5 5 6

Character found at index 5

Enter the character to search (e to exit): B

low mid high

Iteration: 0 4 9

Iteration: 0 1 3

Iteration: 0 0 0

Character not found

Enter the character to search (e to exit):

**BINARY SEARCH**

#include <stdio.h>  
int binarySearchRecursive(char arr[], int low, int high, char target) {

    if (low <= high) {

        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {

            return mid; // Element found, return index

        } else if (arr[mid] < target) {

            return binarySearchRecursive(arr, mid + 1, high, target); // Search in the right half

        } else {

            return binarySearchRecursive(arr, low, mid - 1, target); // Search in the left half

        }

    }

    return -1; // Element not found

}

int main() {

    int n;

    printf("Enter the size of the character array: ");

    scanf("%d", &n);

    char charArray[n];

    printf("Enter the sorted character array:\n");

    for (int i = 0; i < n; i++) {

        scanf(" %c", &charArray[i]);

    }

    char target;

    printf("Enter the character to search: ");

    scanf(" %c", &target);

    int result = binarySearchRecursive(charArray, 0, n - 1, target);

**MINMAX**  
  
#include<stdio.h>

int ar[100];

void minmax(int i, int j, int \*mx, int \*mn)

{

    if (i == j)

    {

        \*mx = ar[i];

        \*mn = ar[i];

        printf(" [low = %d    High = %d ]     Min = %d       Max = %d \n", i, j, \*mn, \*mx);

    }

    else if (i == j - 1)

    {

        if (ar[i] < ar[j])

        {

            \*mn = ar[i];

            \*mx = ar[j];

        }

        else

        {

            \*mn = ar[j];

            \*mx = ar[i];

        }

        printf(" [low = %d    High = %d ]     Min = %d      Max = %d \n", i, j, \*mn, \*mx);

    }

    else

    {

        int mx1, mn1;

        int mid = (i + j) / 2;

        minmax(i, mid, mx, mn);

        minmax(mid + 1, j, &mx1, &mn1);

        if (\*mx < mx1)

        {

            \*mx = mx1;

        }

        if (\*mn > mn1)

        {

            \*mn = mn1;

        }

        printf(" [low = %d     High = %d ]     Min = %d     Max = %d \n", i, j, \*mn, \*mx);

    }

}

int main()

{

    int n;

    printf("Enter the number of elements: ");

    scanf("%d", &n);

    int max, min, i;

    printf("Enter the elements of array: ");

    for (i = 0; i < n; i++)

    {

        scanf("%d", &ar[i]);

    }

    minmax(0, n-1, &max, &min);

    printf("Min = %d   Max = %d", min, max);

    return 0;

}

OUTPUT  
  
PS D:\MADF\_lab\_sem4\output> & .\'minMax.exe'

Enter the number of elements: 4

Enter the elements of array: 1

2

3

4

[low = 0 High = 1 ] Min = 1 Max = 2

[low = 2 High = 3 ] Min = 3 Max = 4

[low = 0 High = 3 ] Min = 1 Max = 4

Min = 1 Max = 4

**QUICKSORT**  
  
#include <stdio.h>

int ar[9];

void interchange(int ar[], int i, int j)

{

    ar[i] = ar[i] + ar[j];

    ar[j] = ar[i] - ar[j];

    ar[i] = ar[i] - ar[j];

}

int partition(int ar[], int m, int p)

{

    int v, i, j;

    v = ar[m];

    i = m;

    j = p;

    do

    {

        do

        {

            i = i + 1;

        } while (ar[i] < v);

        do

        {

            j = j - 1;

        } while (ar[j] > v);

        if (i < j)

        {

            interchange(ar, i, j);

        }

    } while (i < j);

    ar[m] = ar[j];

    ar[j] = v;

    return j;

}

        }

    }

}

void quicksort(int p, int q)

{

    if (p < q)

    {

        int j = partition(ar, p, q + 1);

        quicksort(p, j - 1);

         printf("[");

        for (int i = 0; i < 9; i++)

        {

            printf(" %d ", ar[i]);

        }

        printf("]\n");

        quicksort(j + 1, q);

        printf("[");

        for (int i = 0; i < 9; i++)

        {

            printf(" %d ", ar[i]);

        }

        printf("]\n");

    }

}

int main()

{

    printf("Enter the array elements:\n");

    for (int i = 0; i < 9; i++)

    {

        scanf("%d", &ar[i]);

    }

    int start, end;

    printf("Enter start and end positions: ");

    scanf("%d%d", &start, &end);

    printf("Array after each iteration:\n");

    quicksort(start, end);

    return 0;

}

**Kth SMALLEST ELEMENT**  
#include <stdio.h>

#include <limits.h>

#define infinity INT\_MAX

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

    int t = \*a;

    \*a = \*b;

    \*b = t;

}

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

    int pivot, temp, i, j;

    pivot = arr[low];

    i = low;

    j = high + 1;

    do {

        do {

            i++;

        } while (arr[i] < pivot && i <= high);

        do {

            j--;

        } while (arr[j] > pivot && j >= low);

        if (i < j) {

            swap\_int(&arr[i], &arr[j]);

        }

    } while (i <= j);

    arr[low] = arr[j];

    arr[j] = pivot;

    return j;

}

void select\_int(int a[], int n, int k) {

    int low = 1, up = n;

    do {

        int j = partition\_int(a, low, up);

        printf("pass: ");

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

            if (i == j)

                printf("\e[1m|%d| \e[0m", a[i]);

            else

                printf(" %d ", a[i]);

        }

        printf("j=\t%d\n", j);

void select\_char(char a[], int n, int k) {

    int low = 1, up = n;

    do {

        int j = partition\_char(a, low, up);

        printf("pass: ");

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

            if (i == j)

                printf("\e[1m|%c| \e[0m", a[i]);

            else

                printf(" %c ", a[i]);

        }

        printf("j=\t%d\n", j);

        // Print the partition

        printf("Partition: ");

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

            if (i == j)

                printf("\e[1m|%c| \e[0m", a[i]);

            else

                printf(" %c ", a[i]);

        }

        printf("\n");

        if (k == j) {

            return;

        } else if (k > j)

            low = j + 1;

        else

            up = j;

    } while (low < up);

}

int main() {

    int choice;

    do {

        printf("Select the array type:\n");

        printf("1. Integer array\n");

        printf("2. Character array\n");

        printf("3. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1: {

                int n;

                printf("Enter the size of the array: ");

                scanf("%d", &n);

                int a[n + 1]; // Corrected the array size

                printf("Enter the elements of the array: ");

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

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

                a[n + 1] = infinity;

                int k;

                printf("Enter the value of k to find the kth smallest element: ");

                scanf("%d", &k);

                select\_int(a, n, k);

                printf("The %d smallest element is %d\n", k, a[k]);

                break;

            }

            case 2: {

                int n;

                printf("Enter the size of the array: ");

                scanf("%d", &n);

                char a[n + 1]; // Corrected the array size

                printf("Enter the elements of the array: ");

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

                    scanf(" %c", &a[i]);

                a[n + 1] = infinity;

                int k;

                printf("Enter the value of k to find the kth smallest element: ");

                scanf("%d", &k);

                select\_char(a, n, k);

                printf("The %d smallest element is %c\n", k, a[k]);

                break;

            }

            case 3:

                printf("Exiting...\n");

                break;

            default:

                printf("Invalid choice\n");

        }

    } while (choice != 3);

    return 0;

}

    // Print the partition

        printf("Partition: ");

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

            if (i == j)

                printf("\e[1m|%d| \e[0m", a[i]);

            else

                printf(" %d ", a[i]);

        }

        printf("\n");

        if (k == j) {

            return;

        } else if (k > j)

            low = j + 1;

        else

            up = j;

    } while (low < up);

}

void swap\_char(char \*a, char \*b) {

    char t = \*a;

    \*a = \*b;

    \*b = t;

}

int partition\_char(char arr[], int low, int high) {

    int temp, i, j;

    char pivot = arr[low];

    i = low;

    j = high + 1;

    do {

        do {

            i++;

        } while (arr[i] < pivot && i <= high);

        do {

            j--;

        } while (arr[j] > pivot && j >= low);

        if (i < j) {

            swap\_char(&arr[i], &arr[j]);

        }

    } while (i <= j);

    arr[low] = arr[j];

    arr[j] = pivot;

    return j;

}

void select\_char(char a[], int n, int k) {

    int low = 1, up = n;

    do {

        int j = partition\_char(a, low, up);

        printf("pass: ");

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

            if (i == j)

                printf("\e[1m|%c| \e[0m", a[i]);

            else

                printf(" %c ", a[i]);

        }

        printf("j=\t%d\n", j);

        // Print the partition

        printf("Partition: ");

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

            if (i == j)

                printf("\e[1m|%c| \e[0m", a[i]);

            else

                printf(" %c ", a[i]);

        }

        printf("\n");

        if (k == j) {

            return;

        } else if (k > j)

            low = j + 1;

        else

            up = j;

    } while (low < up);

}

int main() {

    int choice;

    do {

        printf("Select the array type:\n");

        printf("1. Integer array\n");

        printf("2. Character array\n");

        printf("3. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1: {

                int n;

                printf("Enter the size of the array: ");

                scanf("%d", &n);

                int a[n + 1]; // Corrected the array size

                printf("Enter the elements of the array: ");

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

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

                a[n + 1] = infinity;

                int k;

                printf("Enter the value of k to find the kth smallest element: ");

                scanf("%d", &k);

                select\_int(a, n, k);

                printf("The %d smallest element is %d\n", k, a[k]);

                break;

            }

            case 2: {

                int n;

                printf("Enter the size of the array: ");

                scanf("%d", &n);

                char a[n + 1]; // Corrected the array size

                printf("Enter the elements of the array: ");

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

                    scanf(" %c", &a[i]);

                a[n + 1] = infinity;

                int k;

                printf("Enter the value of k to find the kth smallest element: ");

                scanf("%d", &k);

                select\_char(a, n, k);

                printf("The %d smallest element is %c\n", k, a[k]);

                break;

            }

            case 3:

                printf("Exiting...\n");

                break;

            default:

                printf("Invalid choice\n");

        }

    } while (choice != 3);

    return 0;

}

Select the array type:

1. Integer array

2. Character array

3. Exit

Enter your choice: 1

Enter the size of the array: 11

Enter the elements of the array: 53

18

43

29

89

17

55

68

79

67

35

Enter the value of k to find the kth smallest element: 6

pass: 17 18 43 29 35 |53| 55 68 79 67 89 j= 6

Partition: 17 18 43 29 35 |53| 55 68 79 67 89

The 6 smallest element is 53

Select the array type:

1. Integer array

2. Character array

3. Exit

Enter your choice: 2

Enter the size of the array: 12

Enter the elements of the array: Q

M

C

U

A

M

T

F

J

Y

L

G

Enter the value of k to find the kth smallest element: 4

pass: J M C G A M L F |Q| Y T U j= 9

Partition: J M C G A M L F |Q| Y T U

pass: A F C G |J| M L M Q Y T U j= 5

Partition: A F C G |J| M L M Q

pass: |A| F C G J M L M Q Y T U j= 1

Partition: |A| F C G J

pass: A C |F| G J M L M Q Y T U j= 3

Partition: C |F| G J

pass: A C F |G| J M L M Q Y T U j= 4

Partition: |G| J

The 4 smallest element is G

|43| -12 11 58 -5 29 65 -17 37 0 i: 0, j: 10

|-17| -12 11 0 -5 29 37 43 65 58 i: 0, j: 7

-17 |-12| 11 0 -5 29 37 43 65 58 i: 1, j: 7

-17 -12 |11| 0 -5 29 37 43 65 58 i: 2, j: 7

-17 -12 |-5| 0 11 29 37 43 65 58 i: 2, j: 4

-17 -12 -5 0 11 |29| 37 43 65 58 i: 5, j: 7

-17 -12 -5 0 11 29 37 43 |65| 58 i: 8, j: 10

Sorted array: -17 -12 -5 0 11 29 37 43 58 65

R , W , H , A , V , T | M , A , U , D , U ,

R , W , H | A , V , T | M , A , U , D , U ,

R , W | H | A , V , T | M , A , U , D , U ,

R | W | H | A , V , T | M , A , U , D , U ,

R , W | H | A , V , T | M , A , U , D , U ,

H , R , W | A , V , T | M , A , U , D , U ,

H , R , W | A , V | T | M , A , U , D , U ,

H , R , W | A | V | T | M , A , U , D , U ,

H , R , W | A , V | T | M , A , U , D , U ,

H , R , W | A , T , V | M , A , U , D , U ,

A , H , R , T , V , W | M , A , U , D , U ,

A , H , R , T , V , W | M , A , U | D , U ,

A , H , R , T , V , W | M , A | U | D , U ,

A , H , R , T , V , W | M | A | U | D , U ,

A , H , R , T , V , W | A , M | U | D , U ,

A , H , R , T , V , W | A , M , U | D , U ,

A , H , R , T , V , W | A , M , U | D | U ,

A , H , R , T , V , W | A , M , U | D , U ,

A , H , R , T , V , W | A , D , M , U , U ,

A , A , D , H , M , R , T , U , U , V , W ,

**MERGESORT**  
  
#include <stdio.h>

#include <stdlib.h>

#define MAX 20

char sym[MAX];

int size;

void display(char arr[])

{

    for (int i = 0; i < size; i++)

    {

        printf(" %c ", arr[i]);

        if (sym[i] != '\0')

            printf(" %c ", sym[i]);

    }

    printf("\n");

}

void merge(char arr[], char temp[], int low1, int high1, int low2, int high2)

{

    int i = low1, j = low2, k = low1;

    while (i <= high1 && j <= high2)

    {

        if (arr[i] <= arr[j])

            temp[k++] = arr[i++];

        else if (arr[i] > arr[j])

            temp[k++] = arr[j++];

    }

    while (i <= high1)

        temp[k++] = arr[i++];

    while (j <= high2)

        temp[k++] = arr[j++];

    sym[high1] = ',';

}

void copy(char arr[], char temp[], int low, int high)

{

    for (int i = low; i <= high; i++)

        temp[i] = arr[i];

}

void mergeSortRecursive(char arr[], char temp[], int low, int high)

{

    if (low < high)

    {

        int mid = (low + high) / 2;

        sym[mid] = '|';

        display(arr);

        mergeSortRecursive(arr, temp, low, mid);

        mergeSortRecursive(arr, temp, mid + 1, high);

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

        copy(temp, arr, low, high);

        display(arr);

    }

}

void mergeSort(char arr[], int low, int high)

{

    char temp[MAX];

    for (int i = 0; i < MAX; i++)

        sym[i] = ',';

    size = high - low + 1;

    mergeSortRecursive(arr, temp, low, high);

}

int main()

{

    char arr[] = {'R', 'W', 'H', 'A', 'V', 'T', 'M', 'A', 'U', 'D', 'U'};

    int arr\_size = sizeof(arr) / sizeof(arr[0]);

    printf("\n\n");

    mergeSort(arr, 0, arr\_size - 1);

    printf("\n");

    return 0;

}