

1)

```
#include <stdio.h>
```

```
struct pair {
```

```
    int min;
```

```
    int max;
```

```
};
```

```
struct pair minmax(int arr[], int low, int high) {
```

```
    struct Pair result, left, right;
```

```
    if (low == high) {
```

```
        result.min = arr[low];
```

```
        result.max = arr[low];
```

```
        return result;
```

```
    }
```

```
    if (high == low + 1) {
```

```
        if (arr[low] < arr[high]) {
```

```
            result.min = arr[low];
```

```
            result.max = arr[high];
```

```
        }
```

```
    else {
```

```
        result.min = arr[high];
```

```
        result.max = arr[low];
```

```
    }
```

```
    return result;
```

```
}
```

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

```
left = minmax(arr, low, mid);
```

```
right = minmax(arr, mid + 1, high);
```

```
result.min = (left.min < right.min) ? left.min : right.min;
```

```

    result.max = (left.max > right.max) ? left.max : right.max;

    return result;
}

int main() {

    int arr[] = {4,7,9,2,17,48,5};

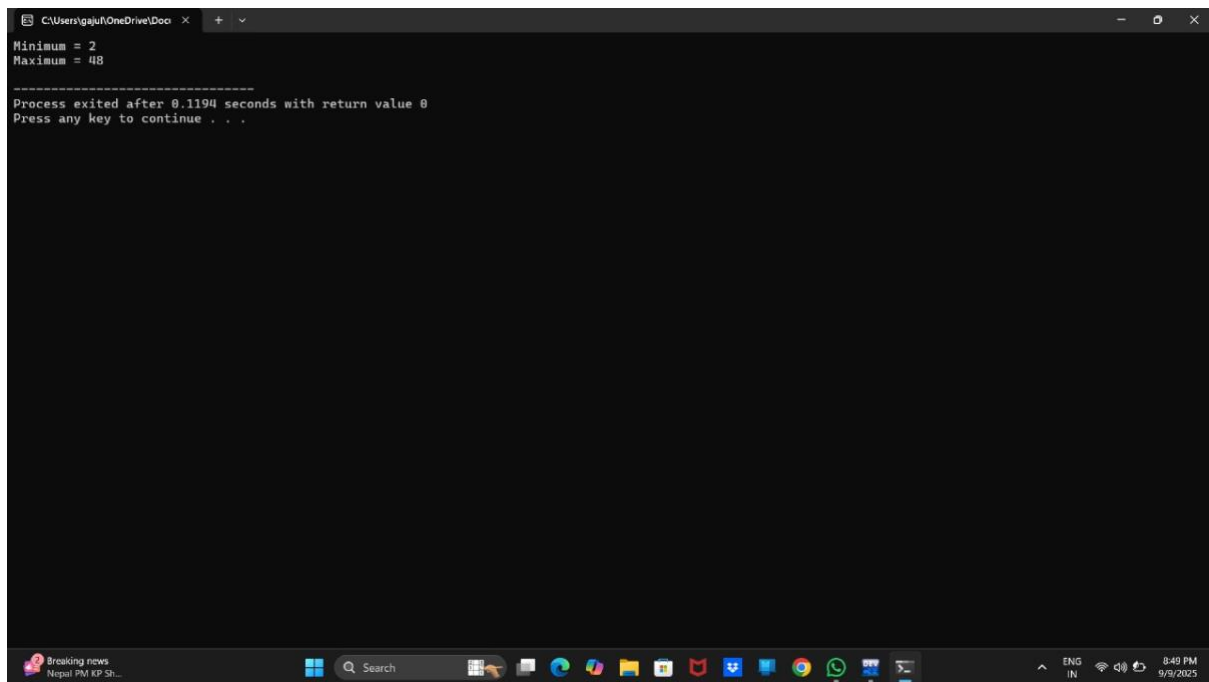
    int n = sizeof(arr) / sizeof(arr[0]);

    struct pair result = fminmax(arr, 0, n - 1);

    printf("Minimum = %d\nMaximum = %d\n", result.min, result.max);

    return 0;
}

```



```

C:\Users\gajul\OneDrive\Doc...
Minimum = 2
Maximum = 48

-----
Process exited after 0.1194 seconds with return value 0
Press any key to continue . . .

```

Divide and conquer reduces the number of comparisons compared to the traditional approach and is also better for large inputs.

2)

```

#include <stdio.h>

#include <limits.h>

int max(int a, int b) {

    return (a > b) ? a : b;

}

```

```

int max3(int a, int b, int c) {
    return max(max(a, b), c);
}

int maxCrossingSum(int arr[], int low, int mid, int high) {
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = mid; i >= low; i--) {
        sum += arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }
    sum = 0;
    int right_sum = INT_MIN;
    for (int i = mid + 1; i <= high; i++) {
        sum += arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }

    return left_sum + right_sum;
}

int maxSubArraySum(int arr[], int low, int high) {
    if (low == high)
        return arr[low];
    int mid = (low + high) / 2;
    return max3(
        maxSubArraySum(arr, low, mid),    // Left subarray
        maxSubArraySum(arr, mid + 1, high), // Right subarray

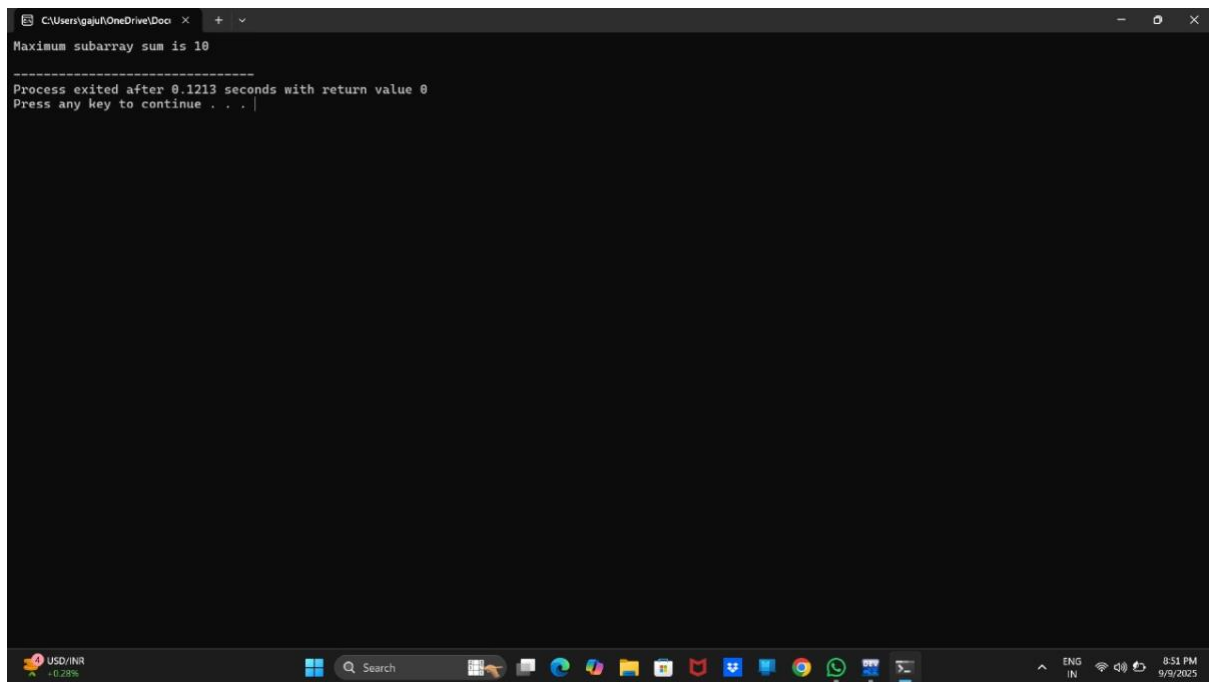
```

```

        maxCrossingSum(arr, low, mid, high)    // Crossing subarray
    );
}

int main() {
    int arr[] = {-2,-1,2,-3,3,1,6,-5};
    int n = sizeof(arr) / sizeof(arr[0]);
    int max_sum = maxSubArraySum(arr, 0, n - 1);
    printf("Maximum subarray sum is %d\n", max_sum);
    return 0;
}

```



```

C:\Users\gajul\OneDrive\Doc
Maximum subarray sum is 10
-----
Process exited after 0.1213 seconds with return value 0
Press any key to continue . . .

```

3)

```

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

struct Node {
    int data;
    struct Node* next;
}

```

```

};

struct Node* hashTable[SIZE];

int hashFunction(int key) {
    return key % SIZE;
}

void insert(int key) {
    int index = hashFunction(key);

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = key;

    newNode->next = hashTable[index];

    hashTable[index] = newNode;

    printf("%d inserted at index %d\n", key, index);
}

void search(int key) {
    int index = hashFunction(key);

    struct Node* temp = hashTable[index];

    while (temp != NULL) {
        if (temp->data == key) {
            printf("%d found at index %d\n", key, index);

            return;
        }

        temp = temp->next;
    }

    printf("%d not found in hash table\n", key);
}

void deleteKey(int key) {
    int index = hashFunction(key);

```

```

struct Node* temp = hashTable[index];
struct Node* prev = NULL;
while (temp != NULL) {
    if (temp->data == key) {
        if (prev == NULL) { // deleting first node
            hashTable[index] = temp->next;
        } else {
            prev->next = temp->next;
        }
        free(temp);
        printf("%d deleted from index %d\n", key, index);
        return;
    }
    prev = temp;
    temp = temp->next;
}
printf("%d not found in hash table\n", key);
}

void display() {
    printf("\nHash Table:\n");
    for (int i = 0; i < SIZE; i++) {
        printf("Index %d: ", i);
        struct Node* temp = hashTable[i];
        if (temp == NULL) {
            printf("NULL");
        }
        while (temp != NULL) {
            printf("%d -> ", temp->data);

```

```

        temp = temp->next;
    }
    printf("\n");
}
}

```

```

int main() {
    int choice, key;
    for (int i = 0; i < SIZE; i++) {
        hashTable[i] = NULL;
    }
    while (1) {
        printf("\n---- Hash Table Operations ----\n");
        printf("1. Insert\n");
        printf("2. Search\n");
        printf("3. Delete\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &key);
                insert(key);
                break;
            case 2:
                printf("Enter value to search: ");

```

```
        scanf("%d", &key);  
        search(key);  
        break;  
case 3:  
    printf("Enter value to delete: ");  
    scanf("%d", &key);  
    deleteKey(key);  
    break;  
case 4:  
    display();  
    break;  
case 5:  
    printf("Exiting\n");  
    exit(0);  
default:  
    printf("Invalid choice! Try again.\n");  
    }  
}  
return 0;  
}
```



```
C:\Users\gajuh\OneDrive\Doc  x + v
16 inserted at index 6

---- Hash Table Operations ----
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice:
1
Enter value to insert: 76
76 inserted at index 6

---- Hash Table Operations ----
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice: 2
Enter value to search: 15
15 not found in hash table

---- Hash Table Operations ----
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice: |
```

4 cm of rain  
Thursday

Search

ENG  
IN

8:55 PM  
9/9/2025