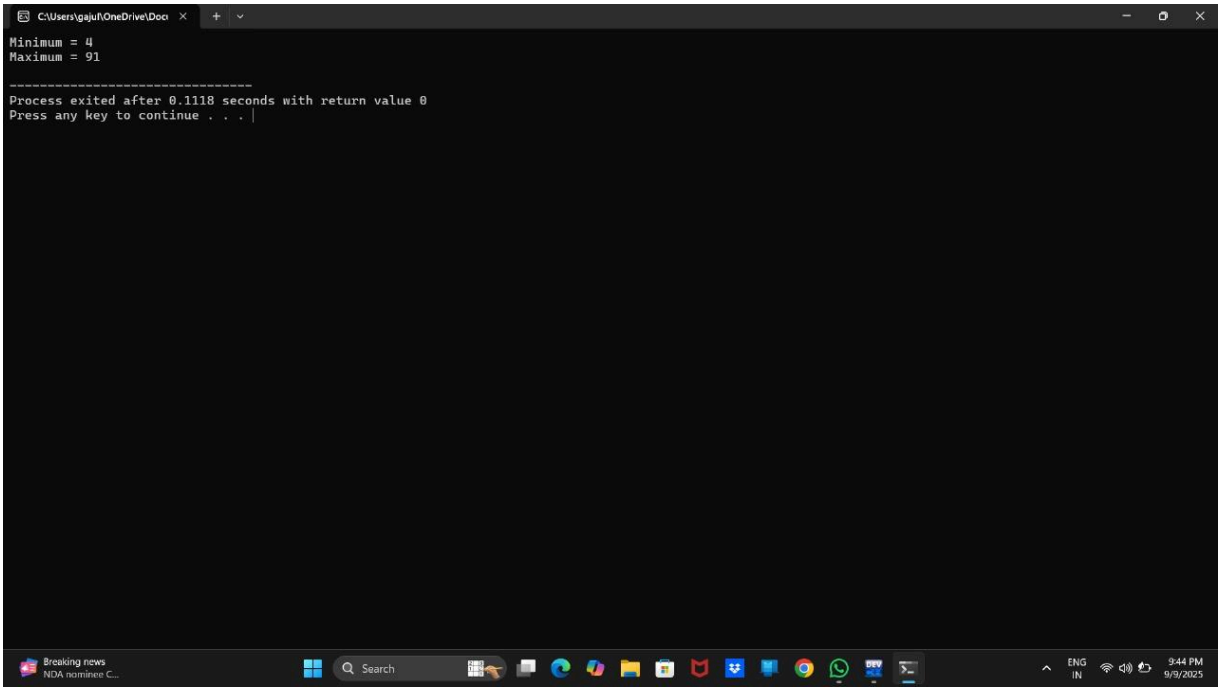
1.Finding minimum and maximum

```
#include <stdio.h>

struct Pair {
    int min;
    int max;
};

struct Pair findMinMax(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 = findMinMax(arr, low, mid);
    right = findMinMax(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[] = {18,6,91,25,17,8,4};
    int n = sizeof(arr) / sizeof(arr[0]);

    struct Pair result = findMinMax(arr, 0, n - 1);

    printf("Minimum = %d\nMaximum = %d\n", result.min, result.max);
    return 0;
```

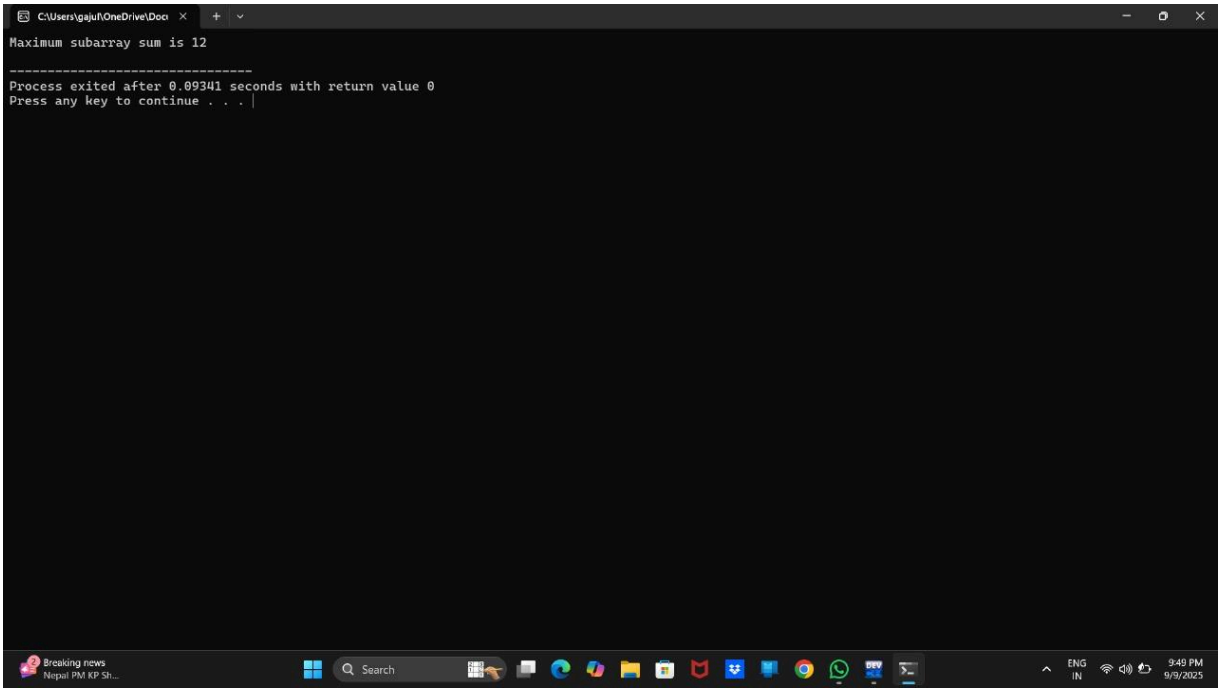
}



2. Maximum subarray

```
#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;
}
```



3.Hash table

```
#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) {
                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;
}
```