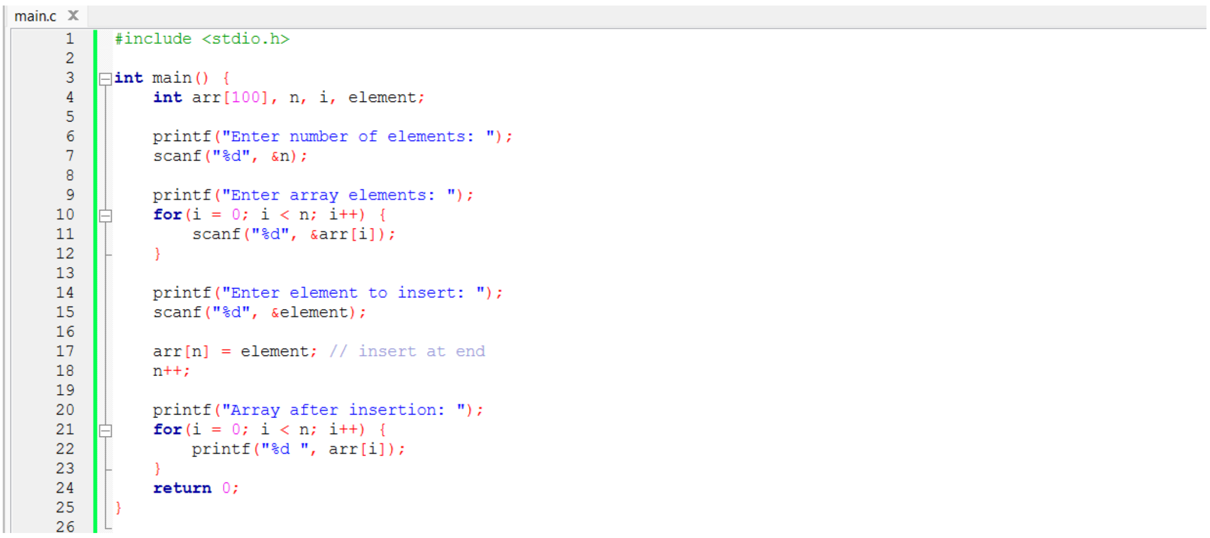
NAME: R S Nithyashree

ROLL NO: 25MCA035

Unordered List

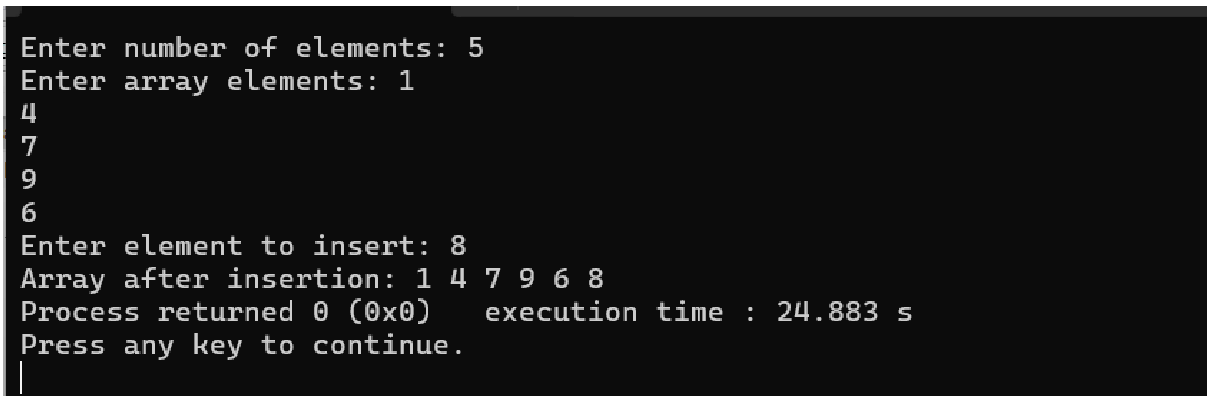
1. Write a c program to insert an element at the end of an unordered array.

PROGRAM:


main.c

```c
#include <stdio.h>

int main() {
    int arr[100], n, i, element;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter array elements: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter element to insert: ");
    scanf("%d", &element);

    arr[n] = element; // insert at end
    n++;

    printf("Array after insertion: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

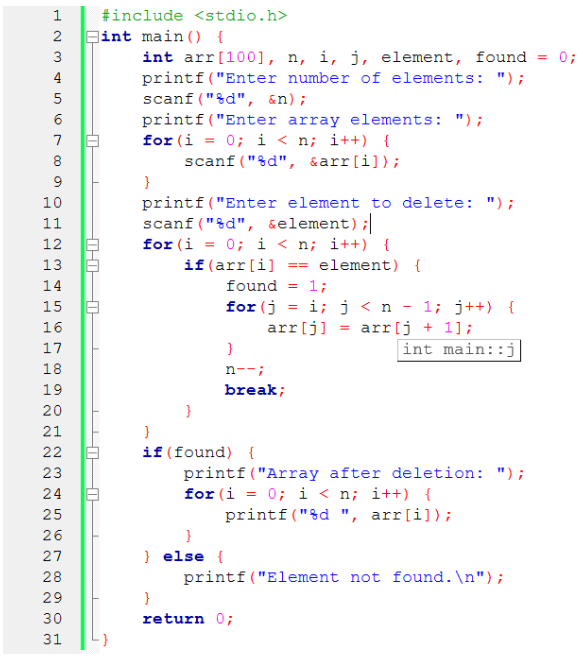
OUTPUT:



```
Enter number of elements: 5
Enter array elements: 1
4
7
9
6
Enter element to insert: 8
Array after insertion: 1 4 7 9 6 8
Process returned 0 (0x0)   execution time : 24.883 s
Press any key to continue.
```

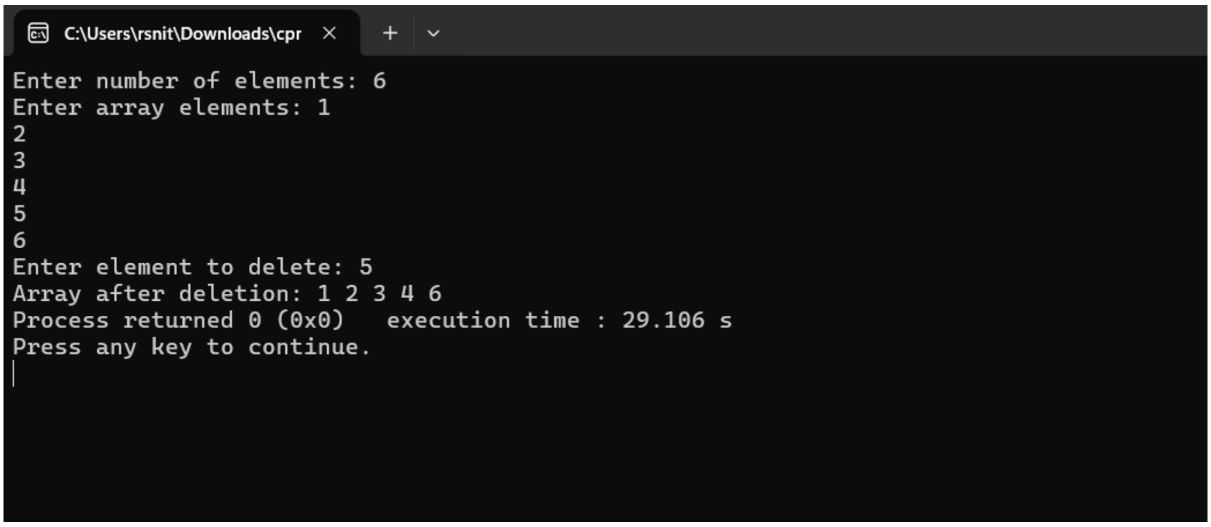
2. Write a c program to delete an element by its value from an unordered array.

PROGRAM:



```c
#include <stdio.h>
int main() {
    int arr[100], n, i, j, element, found = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter array elements: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter element to delete: ");
    scanf("%d", &element);
    for(i = 0; i < n; i++) {
        if(arr[i] == element) {
            found = 1;
            for(j = i; j < n - 1; j++) {
                arr[j] = arr[j + 1];
            }
            n--;
            break;
        }
    }
    if(found) {
        printf("Array after deletion: ");
        for(i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
    } else {
        printf("Element not found.\n");
    }
    return 0;
}
```

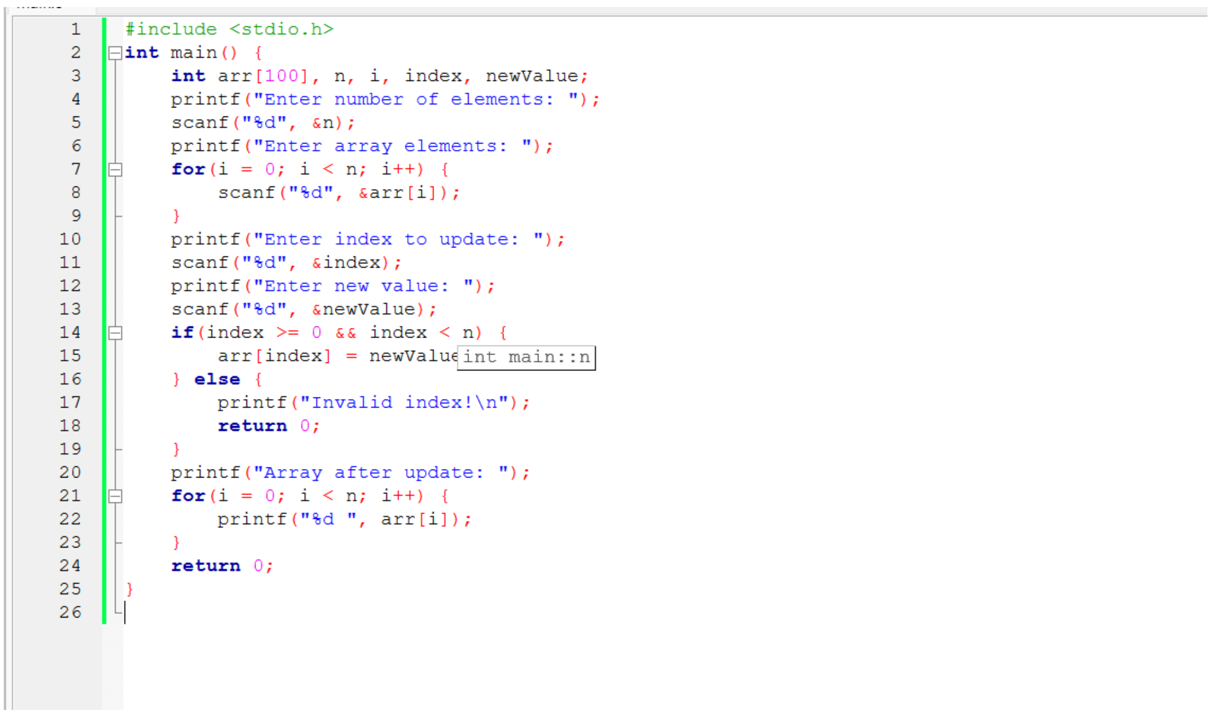
OUTPUT:



```
Enter number of elements: 6
Enter array elements: 1
2
3
4
5
6
Enter element to delete: 5
Array after deletion: 1 2 3 4 6
Process returned 0 (0x0)   execution time : 29.106 s
Press any key to continue.
```

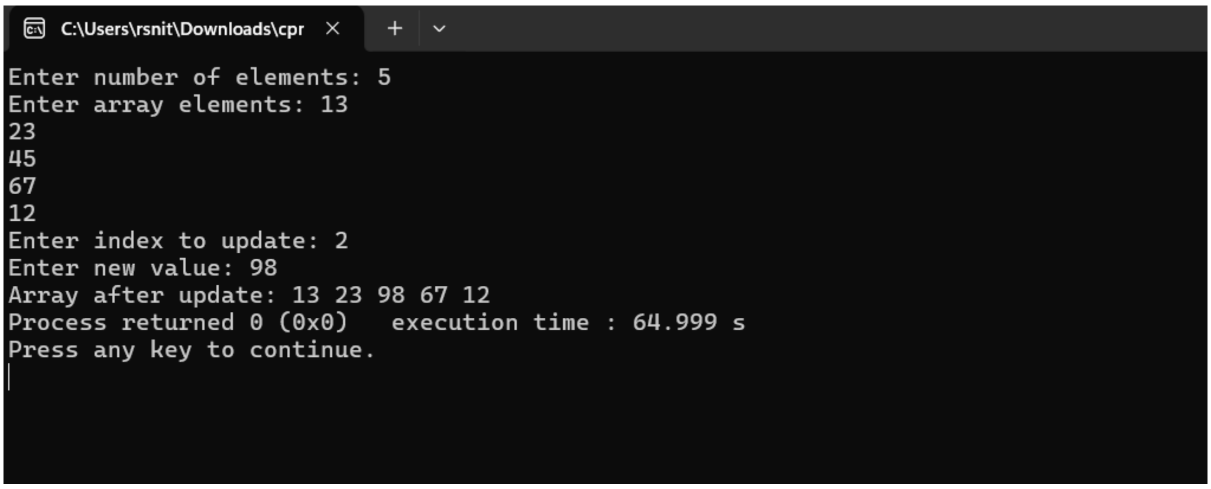
3. Write a c program to update the value of a specific element at a given index in an unordered array.

PROGRAM:



```c
#include <stdio.h>
int main() {
    int arr[100], n, i, index, newValue;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter array elements: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter index to update: ");
    scanf("%d", &index);
    printf("Enter new value: ");
    scanf("%d", &newValue);
    if(index >= 0 && index < n) {
        arr[index] = newValue
    } else {
        printf("Invalid index!\n");
        return 0;
    }
    printf("Array after update: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

OUTPUT:

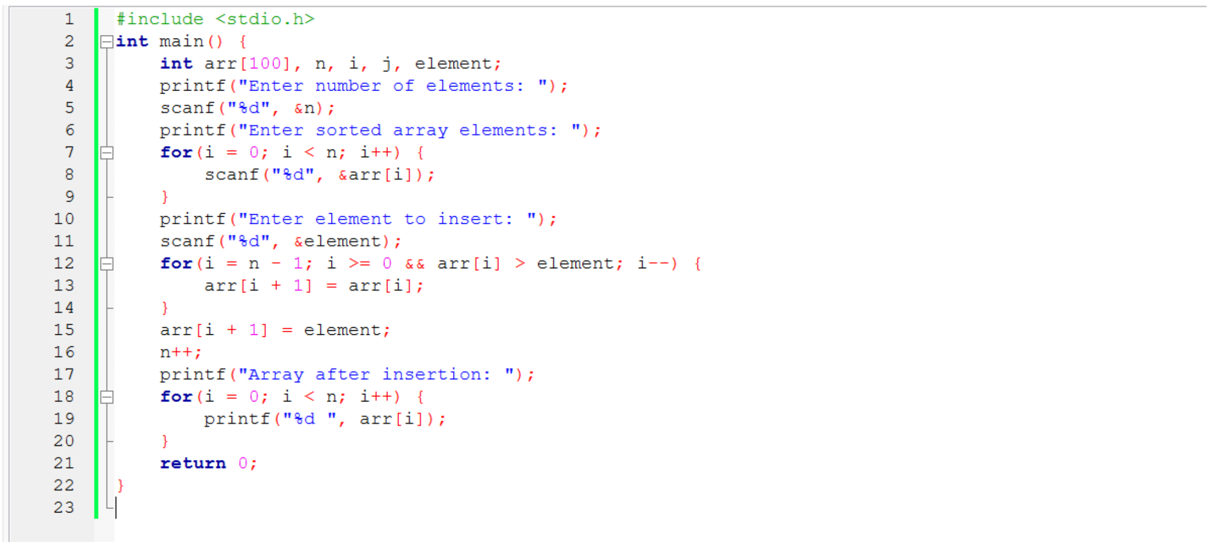


```
Enter number of elements: 5
Enter array elements: 13
23
45
67
12
Enter index to update: 2
Enter new value: 98
Array after update: 13 23 98 67 12
Process returned 0 (0x0)   execution time : 64.999 s
Press any key to continue.
```

Ordered List

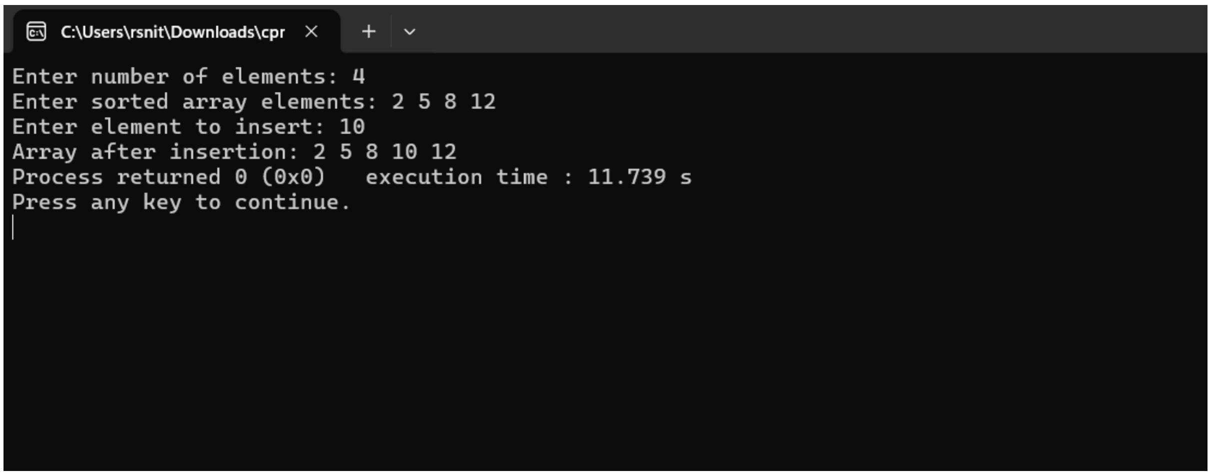
1. Write a c program to insert an element into its correct sorted position in an ordered array.

PROGRAM:



```c
#include <stdio.h>
int main() {
    int arr[100], n, i, j, element;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter sorted array elements: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter element to insert: ");
    scanf("%d", &element);
    for(i = n - 1; i >= 0 && arr[i] > element; i--) {
        arr[i + 1] = arr[i];
    }
    arr[i + 1] = element;
    n++;
    printf("Array after insertion: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}

```

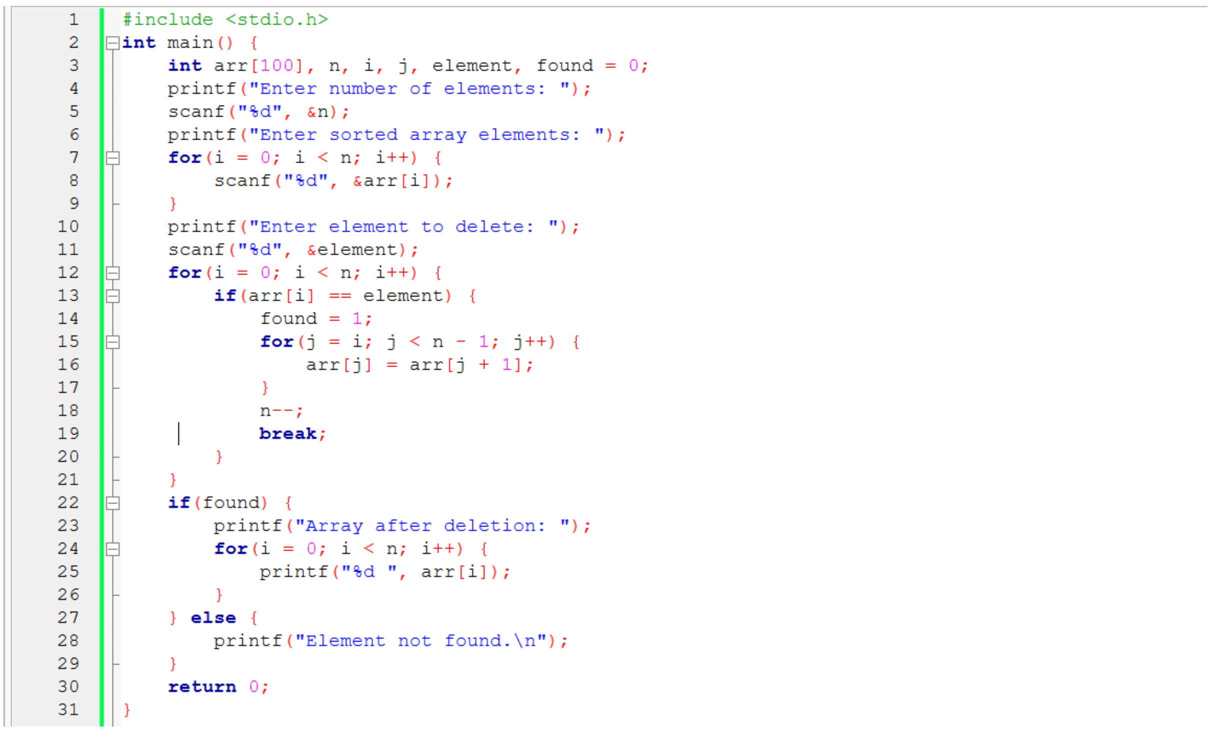
OUTPUT:



```
Enter number of elements: 4
Enter sorted array elements: 2 5 8 12
Enter element to insert: 10
Array after insertion: 2 5 8 10 12
Process returned 0 (0x0)   execution time : 11.739 s
Press any key to continue.
```

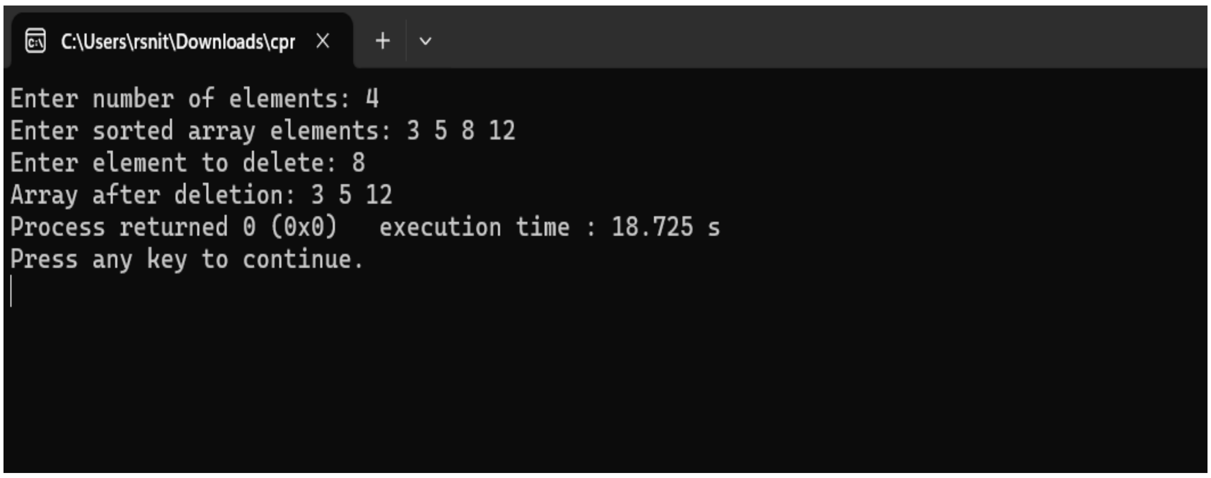
2. Write a C program to delete an element by its value from an ordered array and shift remaining elements accordingly.

PROGRAM:



```c
#include <stdio.h>
int main() {
    int arr[100], n, i, j, element, found = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter sorted array elements: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter element to delete: ");
    scanf("%d", &element);
    for(i = 0; i < n; i++) {
        if(arr[i] == element) {
            found = 1;
            for(j = i; j < n - 1; j++) {
                arr[j] = arr[j + 1];
            }
            n--;
            break;
        }
    }
    if(found) {
        printf("Array after deletion: ");
        for(i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
    } else {
        printf("Element not found.\n");
    }
    return 0;
}
```

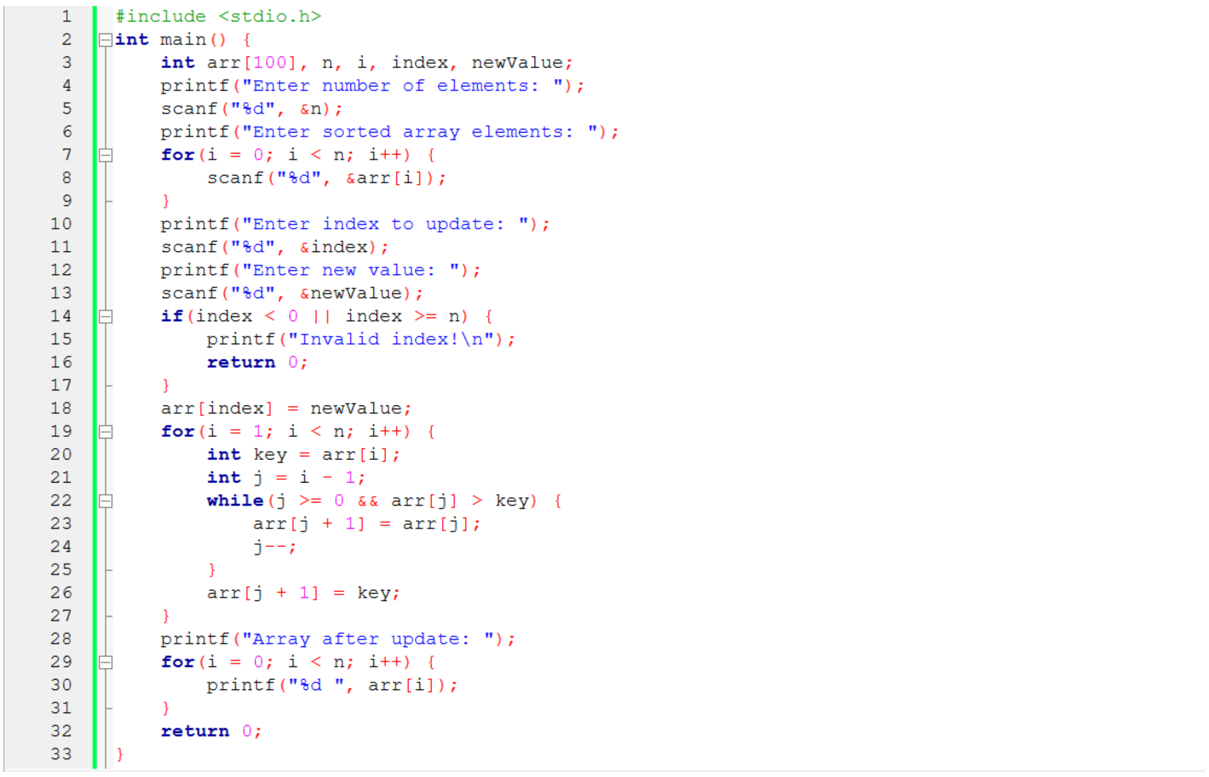
OUTPUT:



```
Enter number of elements: 4
Enter sorted array elements: 3 5 8 12
Enter element to delete: 8
Array after deletion: 3 5 12
Process returned 0 (0x0)   execution time : 18.725 s
Press any key to continue.
```

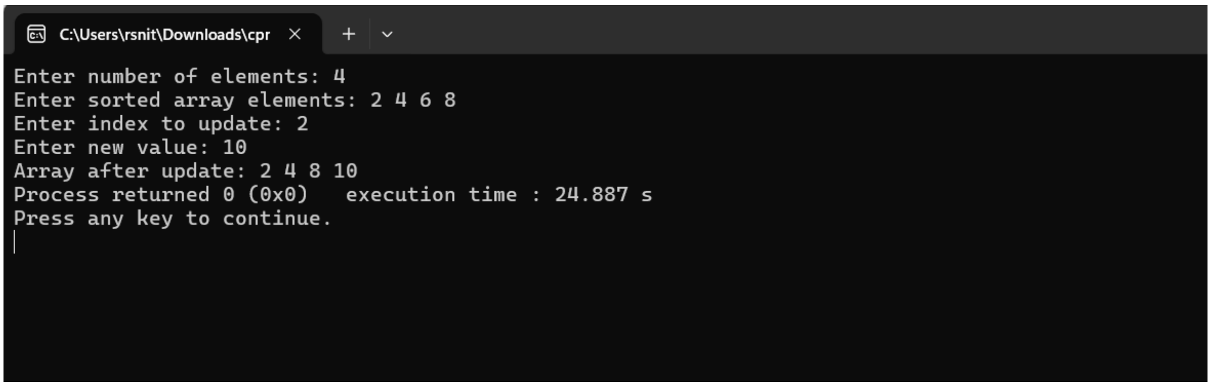
3. Write a C program to update the value of an element and ensure the array remains sorted after the update.

PROGRAM:



```c
#include <stdio.h>
int main() {
    int arr[100], n, i, index, newValue;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter sorted array elements: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter index to update: ");
    scanf("%d", &index);
    printf("Enter new value: ");
    scanf("%d", &newValue);
    if(index < 0 || index >= n) {
        printf("Invalid index!\n");
        return 0;
    }
    arr[index] = newValue;
    for(i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while(j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
    printf("Array after update: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

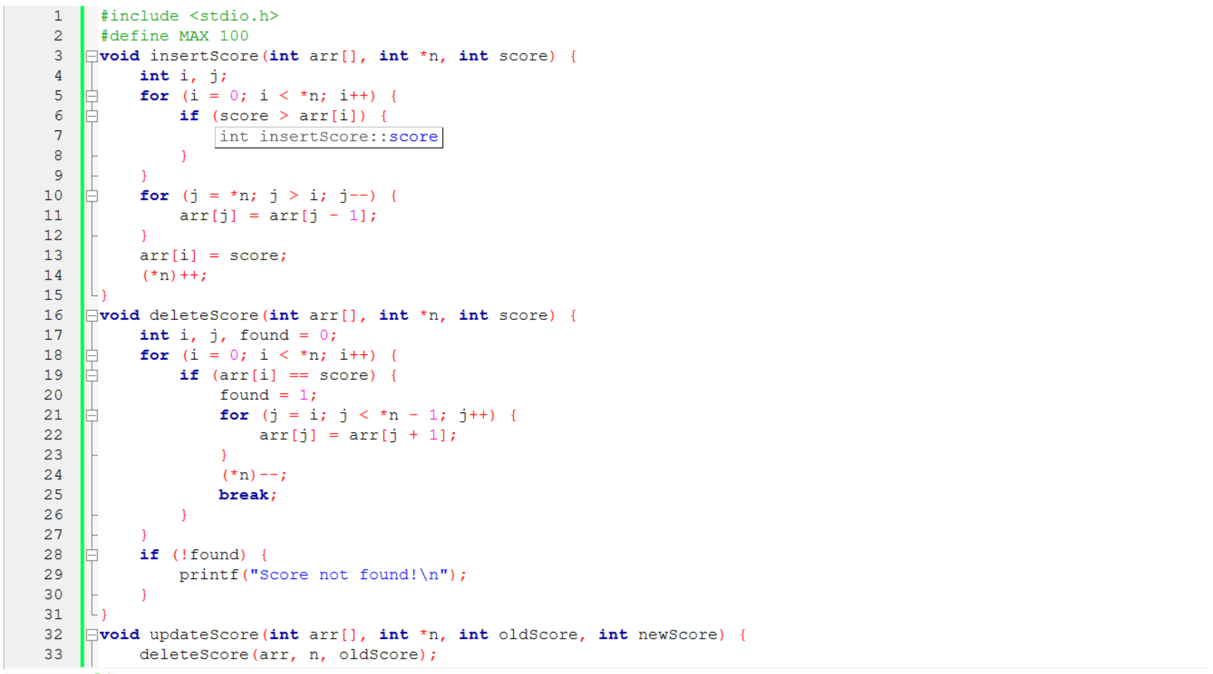
OUTPUT:



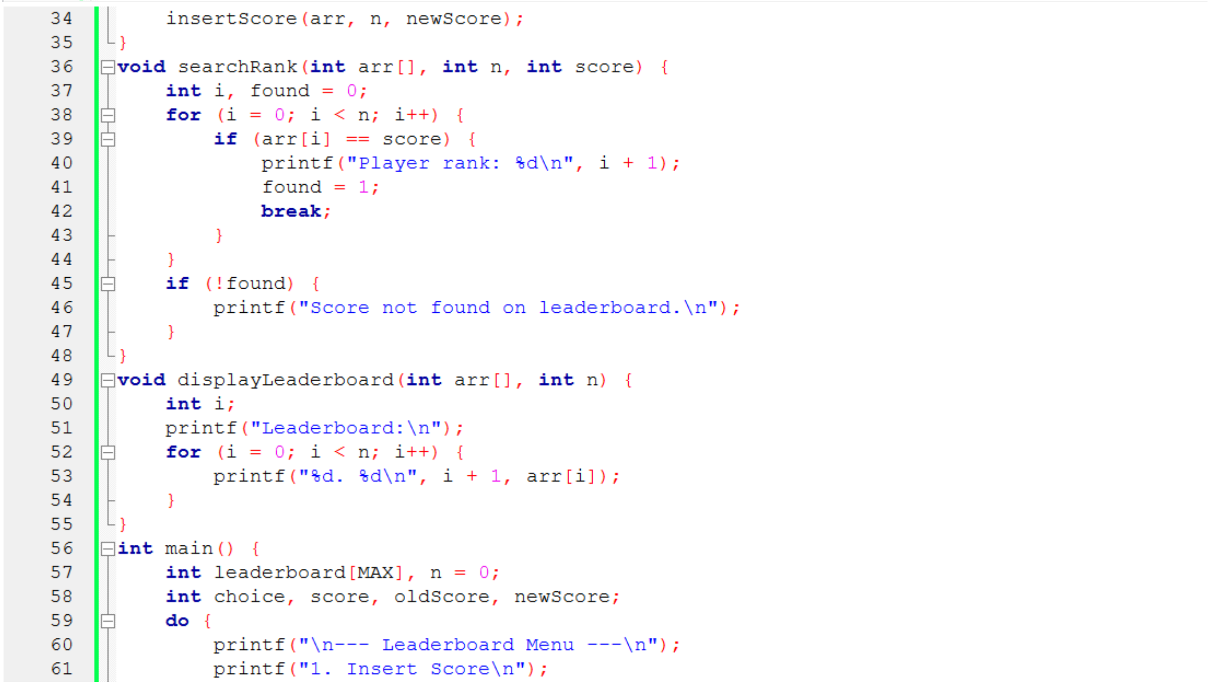
```
Enter number of elements: 4
Enter sorted array elements: 2 4 6 8
Enter index to update: 2
Enter new value: 10
Array after update: 2 4 8 10
Process returned 0 (0x0)   execution time : 24.887 s
Press any key to continue.
```

4. You are maintaining a leaderboard for an online game. The players' scores are stored in descending order, with the highest score first. Write a C program to perform the following tasks.
   4.a) Insertion
   4.b) Deletion
   4.c) Update
   4.d) Search

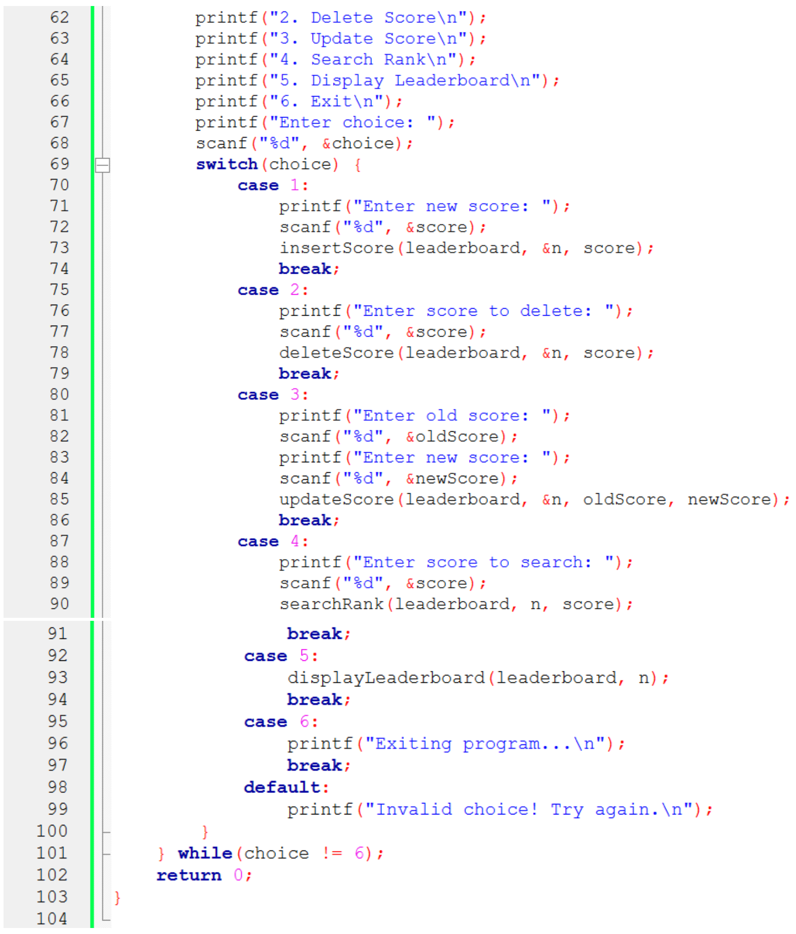
PROGRAM:



```c
#include <stdio.h>
#define MAX 100
void insertScore(int arr[], int *n, int score) {
    int i, j;
    for (i = 0; i < *n; i++) {
        if (score > arr[i]) {

        }
    }
    for (j = *n; j > i; j--) {
        arr[j] = arr[j - 1];
    }
    arr[i] = score;
    (*n)++;
}
void deleteScore(int arr[], int *n, int score) {
    int i, j, found = 0;
    for (i = 0; i < *n; i++) {
        if (arr[i] == score) {
            found = 1;
            for (j = i; j < *n - 1; j++) {
                arr[j] = arr[j + 1];
            }
            (*n)--;
            break;
        }
    }
    if (!found) {
        printf("Score not found!\n");
    }
}
void updateScore(int arr[], int *n, int oldScore, int newScore) {
    deleteScore(arr, n, oldScore);
```

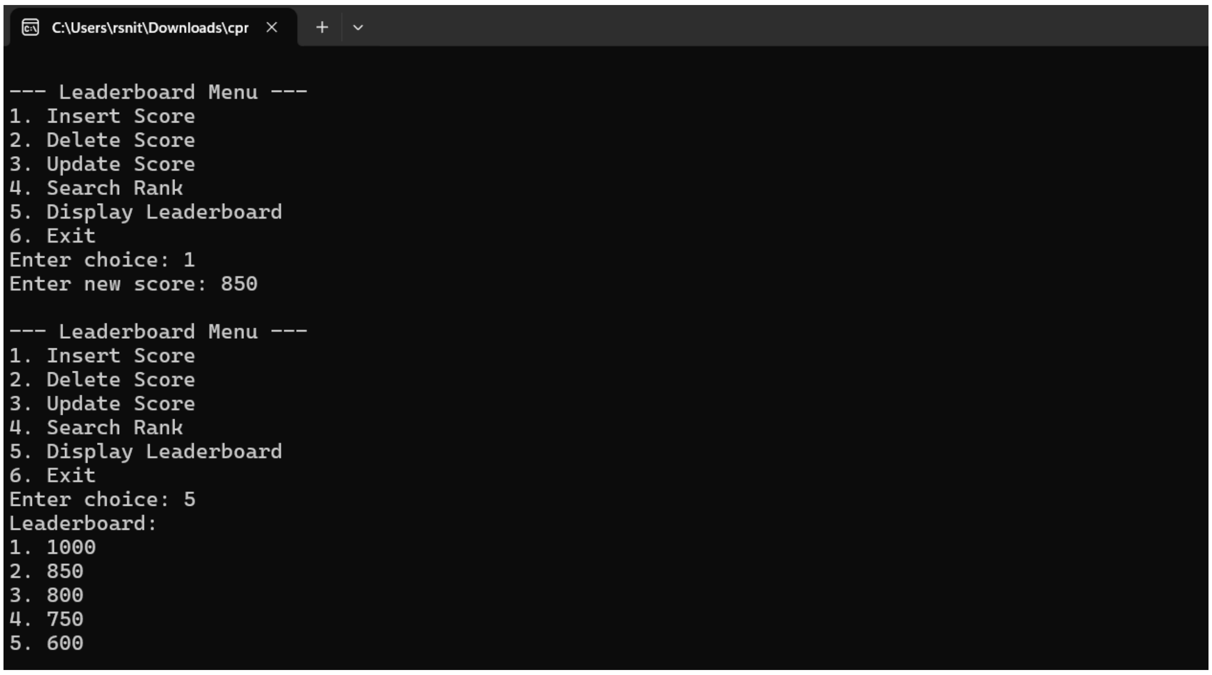


```c
    insertScore(arr, n, newScore);
}
void searchRank(int arr[], int n, int score) {
    int i, found = 0;
    for (i = 0; i < n; i++) {
        if (arr[i] == score) {
            printf("Player rank: %d\n", i + 1);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Score not found on leaderboard.\n");
    }
}
void displayLeaderboard(int arr[], int n) {
    int i;
    printf("Leaderboard:\n");
    for (i = 0; i < n; i++) {
        printf("%d. %d\n", i + 1, arr[i]);
    }
}
int main() {
    int leaderboard[MAX], n = 0;
    int choice, score, oldScore, newScore;
    do {
        printf("\n--- Leaderboard Menu ---\n");
        printf("1. Insert Score\n");
```

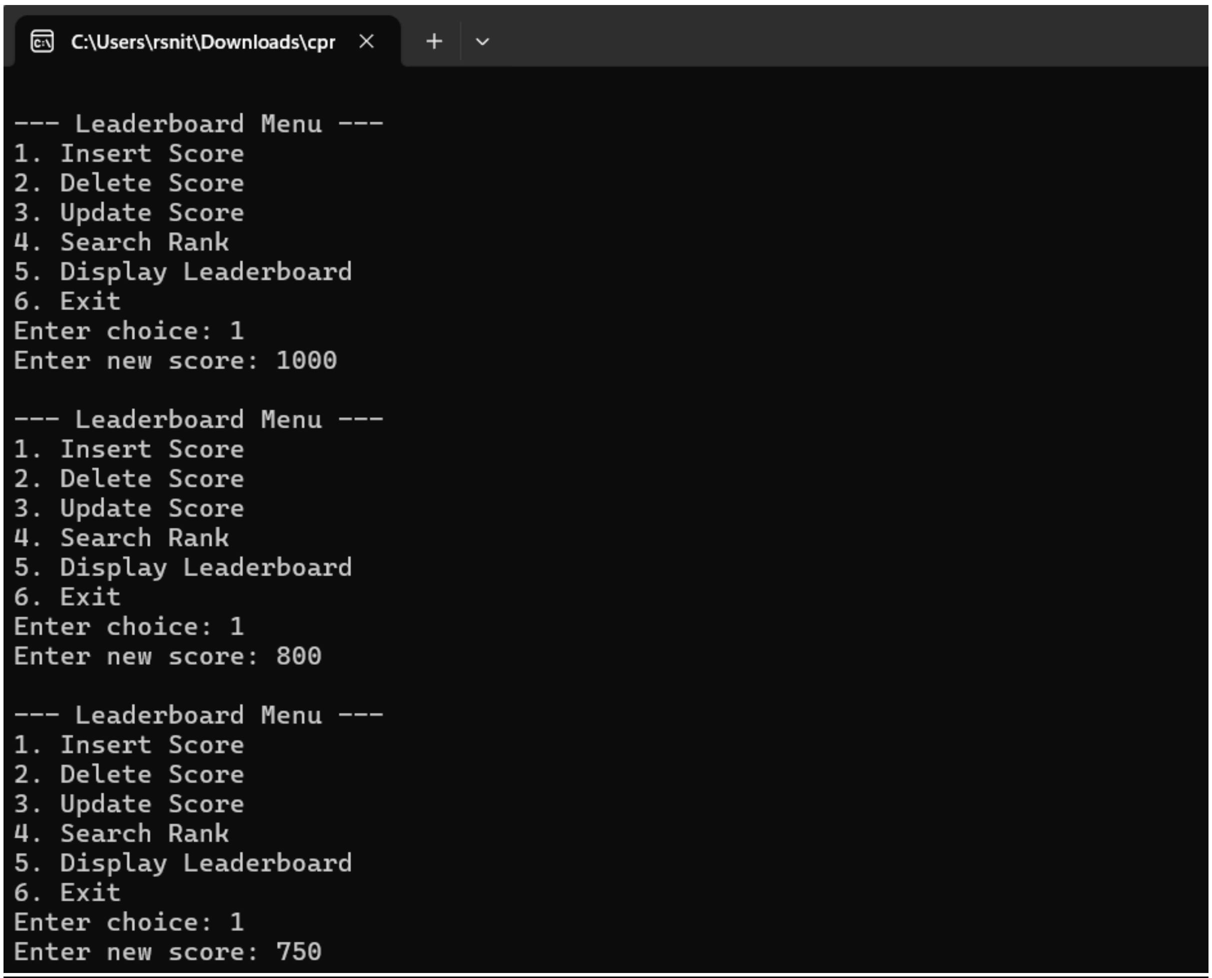


```c
        printf("2. Delete Score\n");
        printf("3. Update Score\n");
        printf("4. Search Rank\n");
        printf("5. Display Leaderboard\n");
        printf("6. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                printf("Enter new score: ");
                scanf("%d", &score);
                insertScore(leaderboard, &n, score);
                break;
            case 2:
                printf("Enter score to delete: ");
                scanf("%d", &score);
                deleteScore(leaderboard, &n, score);
                break;
            case 3:
                printf("Enter old score: ");
                scanf("%d", &oldScore);
                printf("Enter new score: ");
                scanf("%d", &newScore);
                updateScore(leaderboard, &n, oldScore, newScore);
                break;
            case 4:
                printf("Enter score to search: ");
                scanf("%d", &score);
                searchRank(leaderboard, n, score);
                break;
            case 5:
                displayLeaderboard(leaderboard, n);
                break;
            case 6:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice! Try again.\n");
        }
    } while(choice != 6);
    return 0;
}

```

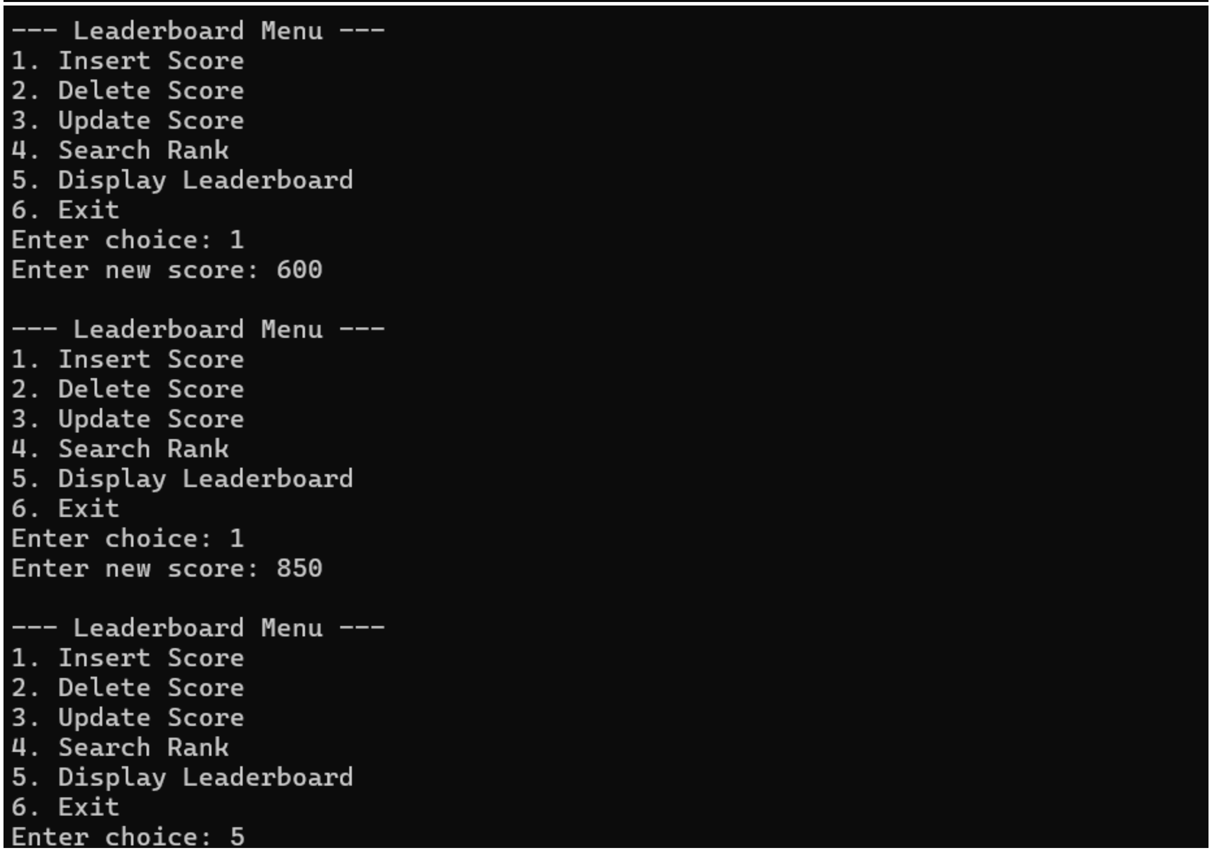
OUTPUT:



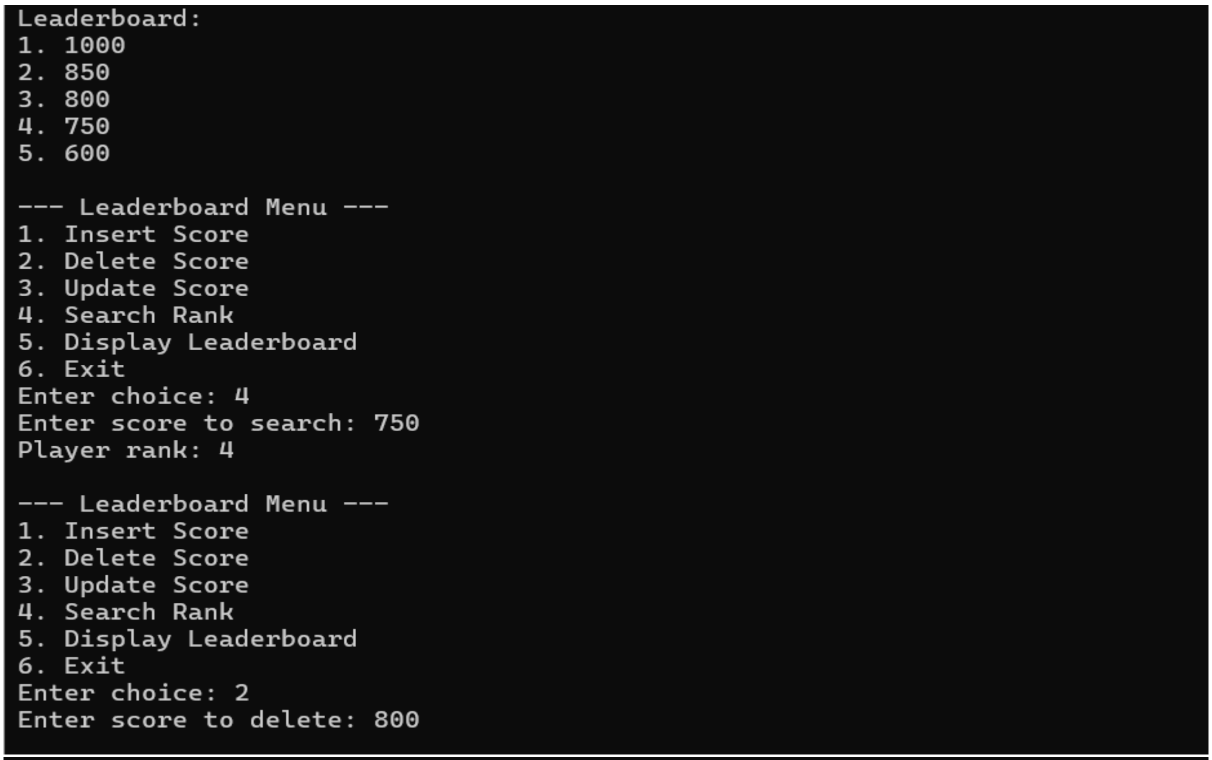
```
--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 1
Enter new score: 850

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 5
Leaderboard:
1. 1000
2. 850
3. 800
4. 750
5. 600
```



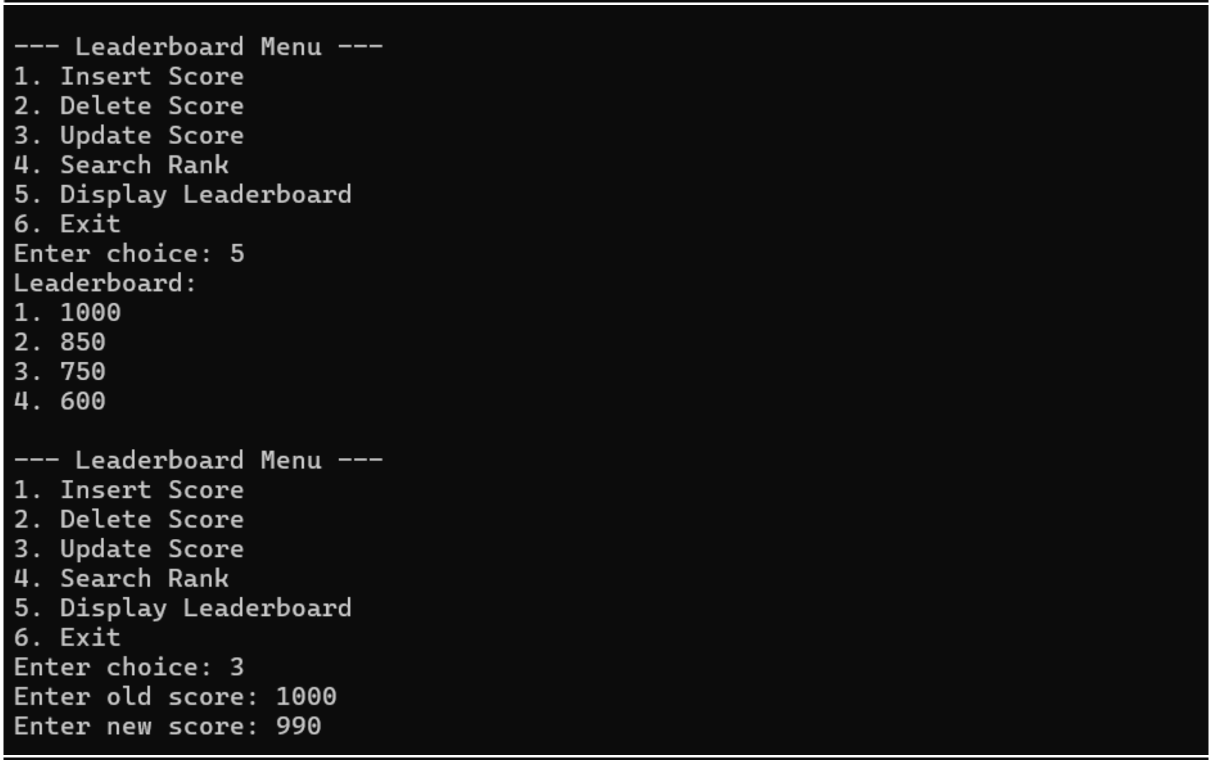
```
--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 1
Enter new score: 1000

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 1
Enter new score: 800

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 1
Enter new score: 750
```



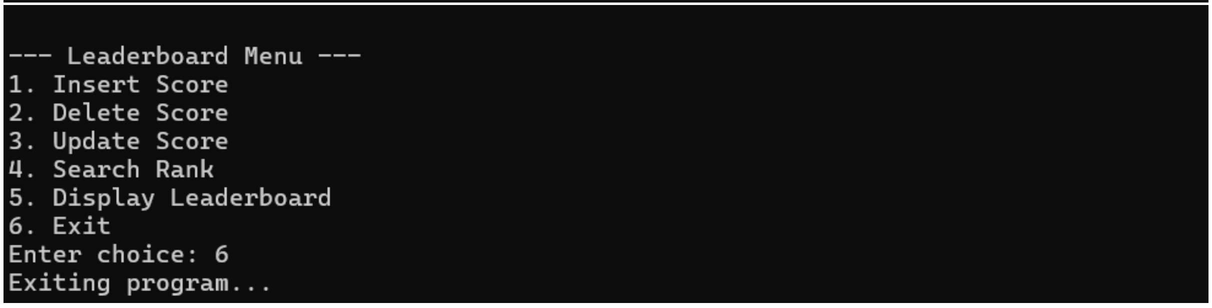
```
--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 1
Enter new score: 600

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 1
Enter new score: 850

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 5
```



```
Leaderboard:
1. 1000
2. 850
3. 800
4. 750
5. 600

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 4
Enter score to search: 750
Player rank: 4

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 2
Enter score to delete: 800
```



```

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 5
Leaderboard:
1. 1000
2. 850
3. 750
4. 600

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 3
Enter old score: 1000
Enter new score: 990
```



```

--- Leaderboard Menu ---
1. Insert Score
2. Delete Score
3. Update Score
4. Search Rank
5. Display Leaderboard
6. Exit
Enter choice: 6
Exiting program...
```