```c
#include <stdio.h>


struct BankAccount {
    int accountNumber;
    char accountHolder[100];
    float balance;
};


void createAccount(struct BankAccount *account) {
    printf("Enter account number: ");
    scanf("%d", &account->accountNumber);

    printf("Enter account holder name: ");
    scanf("%s", account->accountHolder);

    printf("Enter initial balance: ");
    scanf("%f", &account->balance);

    printf("Account created successfully!\n");
}


void withdraw(struct BankAccount *account) {
    float amount;
    printf("Enter withdrawal amount: ");
    scanf("%f", &amount);

    if (amount > 0 && amount <= account->balance) {
        account->balance -= amount;
        printf("Withdrawal successful. New balance: %.2f\n", account->balance);
    } else {
        printf("Invalid withdrawal amount or insufficient balance.\n");
    }
```

```
}


void deposit(struct BankAccount *account) {
    float amount;
    printf("Enter deposit amount: ");
    scanf("%f", &amount);

    if (amount > 0) {
        account->balance += amount;
        printf("Deposit successful. New balance: %.2f\n", account->balance);
    } else {
        printf("Invalid deposit amount.\n");
    }
}


void checkBalance(const struct BankAccount *account) {
    printf("Account Holder: %s\n", account->accountHolder);
    printf("Account Number: %d\n", account->accountNumber);
    printf("Current Balance: %.2f\n", account->balance);
}

int main() {
    struct BankAccount userAccount;
    int choice;

    do {
        printf("\n***** Banking System Menu *****\n");
        printf("1. Create Account\n");
        printf("2. Withdraw\n");
        printf("3. Deposit\n");
        printf("4. Check Balance\n");
        printf("5. Exit\n");
```

```c
do {
    printf("\n***** Banking System Menu *****\n");
    printf("1. Create Account\n");
    printf("2. Withdraw\n");
    printf("3. Deposit\n");
    printf("4. Check Balance\n");
    printf("5. Exit\n");

    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            createAccount(&userAccount);
            break;
        case 2:
            withdraw(&userAccount);
            break;
        case 3:
            deposit(&userAccount);
            break;
        case 4:
            checkBalance(&userAccount);
            break;
        case 5:
            printf("Exiting program. Goodbye!\n");
            break;
        default:
            printf("Invalid choice. Please enter a valid option.\n");
    }
} while (choice != 5);
```

```
***** Banking System Menu *****
1. Create Account
2. Withdraw
3. Deposit
4. Check Balance
5. Exit
Enter your choice: 1
Enter account number: 1BM22CS228
Enter account holder name: Enter initial balance: 25000
Account created successfully!

***** Banking System Menu *****
1. Create Account
2. Withdraw
3. Deposit
4. Check Balance
5. Exit
Enter your choice: 2
Enter withdrawal amount: 2000
Withdrawal successful. New balance: 23000.00

***** Banking System Menu *****
1. Create Account
2. Withdraw
3. Deposit
4. Check Balance
5. Exit
Enter your choice: 3
Enter deposit amount: 4000
Deposit successful. New balance: 27000.00

***** Banking System Menu *****
1. Create Account
2. Withdraw
3. Deposit
4. Check Balance
5. Exit
```

```
Enter your choice: 4
Account Holder: BM22CS228
Account Number: 1
Current Balance: 27000.00

***** Banking System Menu *****
1. Create Account
2. Withdraw
3. Deposit
4. Check Balance
5. Exit
Enter your choice: 5
Exiting program. Goodbye!
```

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

#define MAX_STRINGS 5
#define MAX_LENGTH 50


void swapStrings(char str1[], char str2[]) {
    char temp[MAX_LENGTH];
    strcpy(temp, str1);
    strcpy(str1, str2);
    strcpy(str2, temp);
}


void lexicographicalSort(char strings[][MAX_LENGTH], int n) {
    int i, j;

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            // Compare strings ignoring case
            if (strcasecmp(strings[j], strings[j + 1]) > 0) {
                // Swap strings if they are in the wrong order
                swapStrings(strings[j], strings[j + 1]);
            }
        }
    }
}

int main() {
    char strings[MAX_STRINGS][MAX_LENGTH];
    int i;
```

```
}

int main() {
    char strings[MAX_STRINGS][MAX_LENGTH];
    int i;

    printf("Enter %d strings:\n", MAX_STRINGS);


    for (i = 0; i < MAX_STRINGS; i++) {
        printf("Enter string %d: ", i + 1);
        scanf("%s", strings[i]);
    }


    lexicographicalSort(strings, MAX_STRINGS);


    printf("\nSorted strings:\n");
    for (i = 0; i < MAX_STRINGS; i++) {
        printf("%s\n", strings[i]);
    }

    return 0;
}
```

```
Enter elements of the 2D array (3x3):
Enter element at position (1, 1): 1
Enter element at position (1, 2): 2
Enter element at position (1, 3):
3
Enter element at position (2, 1): 4
Enter element at position (2, 2): 5
Enter element at position (2, 3): 6
Enter element at position (3, 1): 7
Enter element at position (3, 2): 8
Enter element at position (3, 3): 9
Enter the element to check for: 5
5 is present in the 2D array.


...Program finished with exit code 0
Press ENTER to exit console.
```

```c
#include <stdio.h>

#define ROWS 3
#define COLS 3


int isElementPresent(int array[][COLS], int rows, int cols, int target) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (array[i][j] == target) {
                return 1;  // Element found
            }
        }
    }
    return 0;  // Element not found
}

int main() {
    int array[ROWS][COLS];
    int target;


    printf("Enter elements of the 2D array (%dx%d):\n", ROWS, COLS);
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("Enter element at position (%d, %d): ", i + 1, j + 1);
            scanf("%d", &array[i][j]);
        }
    }


    printf("Enter the element to check for: ");
    scanf("%d", &target);
```

```c
    printf("Enter elements of the 2D array (%dx%d):\n", ROWS, COLS);
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("Enter element at position (%d, %d): ", i + 1, j + 1);
            scanf("%d", &array[i][j]);
        }
    }


    printf("Enter the element to check for: ");
    scanf("%d", &target);

    if (isElementPresent(array, ROWS, COLS, target)) {
        printf("%d is present in the 2D array.\n", target);
    } else {
        printf("%d is not present in the 2D array.\n", target);
    }

    return 0;
}
```

```
Enter the larger string: I AM A SINGER
Enter the substring to search for: AM
Substring found at index 2 in the larger string.
```

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


int searchSubstring(const char *largerString, const char *substring) {
    int lenLarger = strlen(largerString);
    int lenSubstring = strlen(substring);

    for (int i = 0; i <= lenLarger - lenSubstring; i++) {
        int j;
        for (j = 0; j < lenSubstring; j++) {
            if (largerString[i + j] != substring[j]) {
                break;
            }
        }
        if (j == lenSubstring) {
            return i;
        }
    }

    return -1;
}

int main() {
    char largerString[100];
    char substring[50];

    // Input the larger string
    printf("Enter the larger string: ");
    fgets(largerString, sizeof(largerString), stdin);
    largerString[strcspn(largerString, "\n")] = '\0';

    // Input the substring to search for
    printf("Enter the substring to search for: ");
    fgets(substring, sizeof(substring), stdin);
```

```c
    printf("Enter the larger string: ");
    fgets(largerString, sizeof(largerString), stdin);
    largerString[strcspn(largerString, "\n")] = '\0';


    printf("Enter the substring to search for: ");
    fgets(substring, sizeof(substring), stdin);
    substring[strcspn(substring, "\n")] = '\0';


    int index = searchSubstring(largerString, substring);


    if (index != -1) {
        printf("Substring found at index %d in the larger string.\n", index);
    } else {
        printf("Substring not found in the larger string.\n");
    }

    return 0;
}
```

```
Enter the size of the array: 5
Enter 5 elements into the array:
Enter element 1: 12
Enter element 2: 13
Enter element 3: 14
Enter element 4: 1
Enter element 5: 2
Enter the number to find the last occurrence: 2
The last occurrence of 2 is at index 4.
```

```c
#include <stdio.h>


int lastIndexOccurrence(int array[], int size, int target) {
    int lastIndex = -1;

    for (int i = 0; i < size; i++) {
        if (array[i] == target) {
            lastIndex = i;
        }
    }

    return lastIndex;
}

int main() {
    int size, target;


    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int array[size];


    printf("Enter %d elements into the array:\n", size);
    for (int i = 0; i < size; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
    }


    printf("Enter the number to find the last occurrence: ");
    scanf("%d", &target);
```

```c
    printf("Enter %d elements into the array:\n", size);
    for (int i = 0; i < size; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
    }


    printf("Enter the number to find the last occurrence: ");
    scanf("%d", &target);


    int lastIndex = lastIndexOccurrence(array, size, target);


    if (lastIndex != -1) {
        printf("The last occurrence of %d is at index %d.\n", target, lastIndex);
    } else {
        printf("%d is not found in the array.\n", target);
    }

    return 0;
}
```

```
Enter element 1: 15
Enter element 2: 13
Enter element 3: 12
Enter element 4: 11
Enter the number to search for: 12
12 is found at index 2 in the array.
```

```c
#include <stdio.h>


int linearSearch(int array[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (array[i] == target) {
            return i;
        }
    }
    return -1;
}

int main() {
    int size, target;


    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int array[size];


    printf("Enter %d elements into the array:\n", size);
    for (int i = 0; i < size; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
    }
```

```c
    printf("Enter %d elements into the array:\n", size);
    for (int i = 0; i < size; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
    }


    printf("Enter the number to search for: ");
    scanf("%d", &target);


    int index = linearSearch(array, size, target);

    if (index != -1) {
        printf("%d is found at index %d in the array.\n", target, index);
    } else {
        printf("%d is not found in the array.\n", target);
    }

    return 0;
}
```

```
Enter the size of the sorted array: 4
Enter 4 sorted elements into the array:
Enter element 1: 12
Enter element 2: 13
Enter element 3: 14
Enter element 4: 15
Enter the number to search for: 13
13 is found at index 1 in the sorted array.
```