DS – LAB 1 (7th Dec 2023)

Rishabh Kumar (1BM22CS221)

1. Develop a C program that simulates a basic banking system with functionalities like account creation, withdrawal, deposit, and balance inquiry. Write different user-defined function for each.

```
#include <stdio.h>

// Function to create an account
void createAccount(float *balance) {
    printf("Enter initial deposit amount: ");
    scanf("%f", balance);
    printf("Account created successfully!\n");
}

// Function to withdraw money
void withdraw(float *balance, float amount) {
    if (*balance >= amount) {
        *balance -= amount;
        printf("Withdrawal successful. Remaining balance: %.2f\n", *balance);
    } else {
        printf("Insufficient funds\n");
    }
}

// Function to deposit money
void deposit(float *balance, float amount) {
    *balance += amount;
    printf("Deposit successful. New balance: %.2f\n", *balance);
}
```

```
// Function to check balance
void checkBalance(float balance) {
    printf("Current balance: %.2f\n", balance);
}

int main() {
    float balance = 0;
    int choice;
    float amount;

    do {
        printf("\n1. Create Account\n2. Withdraw\n3. Deposit\n4. Check Balance\n0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createAccount(&balance);
                break;
            case 2:
                printf("Enter withdrawal amount: ");
                scanf("%f", &amount);
                withdraw(&balance, amount);
                break;
            case 3:
                printf("Enter deposit amount: ");
                scanf("%f", &amount);
                deposit(&balance, amount);
                break;
            case 4:
```

```
            checkBalance(balance);
            break;
        case 0:
            printf("Exiting program\n");
            break;
        default:
            printf("Invalid choice\n");
    }

} while (choice != 0);

return 0;
}
```

2. Implement a C program that sorts strings lexicographically, considering uppercase and lowercase letters, and without using the standard library sorting functions.

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

void lexicographicalSort(char *str[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (strcmp(str[i], str[j]) > 0) {
                char *temp = str[i];
                str[i] = str[j];
                str[j] = temp;
            }
        }
    }
```

```
}

int main() {
    int n;
    printf("Enter the number of strings: ");
    scanf("%d", &n);

    char *str[n];
    printf("Enter %d strings:\n", n);
    for (int i = 0; i < n; i++) {
        str[i] = (char *)malloc(100 * sizeof(char));
        scanf("%s", str[i]);
    }

    lexicographicalSort(str, n);

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

    for (int i = 0; i < n; i++) {
        free(str[i]);
    }

    return 0;
}
```

3. Implement a C program to check if a given element is present in a 2D array with a user defined function.

```c
#include <stdio.h>

int isElementPresent(int arr[][3], int rows, int cols, int element) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (arr[i][j] == element) {
                return 1; // Element found
            }
        }
    }
    return 0; // Element not found
}

int main() {
    int rows = 2, cols = 3;
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}};
    int element;

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

    if (isElementPresent(arr, rows, cols, element)) {
        printf("Element %d is present in the 2D array.\n", element);
    } else {
        printf("Element %d is not present in the 2D array.\n", element);
    }

    return 0;
}
```

4. Create a program in C to search for a substring within a larger string with a user defined function.

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

int searchSubstring(char *str, char *substring) {
    if (strstr(str, substring) != NULL) {
        return 1; // Substring found
    }
    return 0; // Substring not found
}

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

    printf("Enter the larger string: ");
    gets(largerString); // Note: gets is used for simplicity, but it's not recommended for actual use
    printf("Enter the substring to search: ");
    gets(substring);

    if (searchSubstring(largerString, substring)) {
        printf("Substring found in the larger string.\n");
    } else {
        printf("Substring not found in the larger string.\n");
    }

    return 0;
}
```

5. Write a C program to find the index of the last occurrence of a number in an array with a user defined function.

```
#include <stdio.h>

int lastOccurrence(int arr[], int size, int num) {
    for (int i = size - 1; i >= 0; i--) {
        if (arr[i] == num) {
            return i; // Last occurrence found
        }
    }
    return -1; // Number not found
}

int main() {
    int size = 5;
    int arr[] = {1, 2, 3, 4, 3};
    int num;

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

    int index = lastOccurrence(arr, size, num);

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

```
    return 0;
}
```

6. Write a C program to search for a specific element in an array using linear search with a user defined function.

```
#include <stdio.h>

int linearSearch(int arr[], int size, int num) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == num) {
            return i; // Element found
        }
    }
    return -1; // Element not found
}

int main() {
    int size = 5;
    int arr[] = {1, 2, 3, 4, 5};
    int num;

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

    int index = linearSearch(arr, size, num);

    if (index != -1) {
        printf("%d found at index %d.\n", num, index);
    } else {
```

```
        printf("%d not found in the array.\n", num);
    }

    return 0;
}
```

7. Implement a C program to perform a binary search on a sorted array with a user defined fuction.

```
#include <stdio.h>

int binarySearch(int arr[], int size, int num) {
    int low = 0, high = size - 1;

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

        if (arr[mid] == num) {
            return mid; // Element found
        } else if (arr[mid] < num) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return -1; // Element not found
}

int main() {
```

```
    int size = 5;
    int arr[] = {1, 2, 3, 4, 5};
    int num;

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

    int index = binarySearch(arr, size, num);

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

    return 0;
}
```

8. Create a program in C to search for the minimum and maximum elements in an array with a user defined function.

```
#include <stdio.h>

void findMinMax(int arr[], int size, int *min, int *max) {
    *min = *max = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] < *min) {
            *min = arr[i];
        } else if (arr[i] > *max) {
```

```
            *max = arr[i];
        }
    }
}

int main() {
    int size = 5;
    int arr[] = {3, 1, 4, 5, 2};
    int min, max;

    findMinMax(arr, size, &min, &max);

    printf("Minimum element: %d\n", min);
    printf("Maximum element: %d\n", max);

    return 0;
}
```