Batch: D-2	**Roll No: 16010122323**
Experiment No. 1	
Grade: AA / AB / BB / BC / CC / CD /DD	

Title: Implementation of Abstract Data Type

Objective: Implementation of ADT without using any standard library function

Expected Outcome of Experiment:

CO	Outcome
CO 1	Explain the different data structures used in problem solving.

Books/ Journals/ Websites referred:

Abstract:-

(Define ADT. Why are they important in data structures?)

ADT stands for Abstract Data Type. It is a theoretical concept used in computer science and programming to define a data type based on its behavior and operations rather than its implementation details.

Abstract Data Types are important in data structures because they:

1. Abstract the behavior and properties, simplifying programming.

2. Encapsulate data and operations, ensuring data integrity.

3. Promote modularity and reusability, enhancing code organization.

4. Focus on high-level aspects, aiding algorithm development.

5. Provide a standard interface, ensuring consistency and interoperability.

6. Allow data structure independence, enabling optimizations and changes.

Abstract Data Type for String

[for chosen data type write value definition and operator definition)

CODE:

```
#include <stdio.h>

#include <string.h>


#define MAX_STR_LEN 100


typedef struct {

    char data[MAX_STR_LEN];

    int len;

} Str;
```

```
Str create() {
    Str s;
    s.data[0] = '\0';
    s.len = 0;
    return s;
}

Str from_cstr(const char *cstr) {
    Str s;
    int i = 0;
    while (cstr[i] != '\0' && i < MAX_STR_LEN - 1) {
        s.data[i] = cstr[i];
        i++;
    }
    s.data[i] = '\0';
    s.len = i;
    return s;
}

Str copy(const Str *src) {
    Str dest;
    int i = 0;
    while (src->data[i] != '\0' && i < MAX_STR_LEN - 1) {
        dest.data[i] = src->data[i];
        i++;
    }
    dest.data[i] = '\0';
    dest.len = i;
    return dest;
}

int compare(const Str *s1, const Str *s2) {
    int i = 0;
    while (s1->data[i] != '\0' && s2->data[i] != '\0') {
        if (s1->data[i] != s2->data[i]) {
            return (s1->data[i] - s2->data[i]);
        }
        i++;
    }
    return (s1->data[i] - s2->data[i]);
}

Str concat(const Str *s1, const Str *s2) {
    Str result;
    int i = 0;
    int j = 0;
    while (s1->data[i] != '\0' && i < MAX_STR_LEN - 1) {
        result.data[i] = s1->data[i];
        i++;
    }
    while (s2->data[j] != '\0' && i < MAX_STR_LEN - 1) {
        result.data[i] = s2->data[j];
        i++;
        j++;
    }
    result.data[i] = '\0';
    result.len = i;
    return result;
}

void print_menu() {
    printf("\nMenu:\n");
    printf("1. Find length of strings\n");
    printf("2. Copy the strings\n");
    printf("3. Compare the strings\n");
    printf("4. Concatenate the strings\n");
    printf("5. Exit\n");
}

int main() {
    char str1[MAX_STR_LEN];
    char str2[MAX_STR_LEN];

    printf("Enter the first string: ");
```

```
fgets(str1, sizeof(str1), stdin);
sscanf(str1, "%[^\n]", str1);

printf("Enter the second string: ");
fgets(str2, sizeof(str2), stdin);
sscanf(str2, "%[^\n]", str2);

Str s1 = from_cstr(str1);
Str s2 = from_cstr(str2);

int choice;
do {
  print_menu();
  printf("Enter your choice: ");
  scanf("%d", &choice);
  getchar(); // Consume the newline character from the previous input

  switch (choice) {
    case 1:
      printf("Length of String 1: %d\n", s1.len);
      printf("Length of String 2: %d\n", s2.len);
      break;
    case 2:
      s1 = copy(&s2);
      printf("String 1 has been copied from String 2: %s\n", s1.data);
```

}

```
                break;
            case 3:
                if (compare(&s1, &s2) == 0) {
                    printf("String 1 and String 2 are equal.\n");
                } else {
                    printf("String 1 and String 2 are not equal.\n");
                }
                break;
            case 4:
                Str concat_str = concat(&s1, &s2);
                printf("Concatenated String: %s\n", concat_str.data);
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please select a valid option.\n");
        }
    } while (choice != 5);

    return 0;
}
```

Implementation Details:

1. Enlist all the Steps followed and various options explored
2. Explain your program logic and methods used.
3. Explain the Importance of the approach followed by you

1. Steps followed and various options explored:

- Defined an Abstract Data Type (ADT) for Strings using a struct with a character array for data and an integer for length.

- Implemented functions to create an empty string, create a string from a C-style string, copy a string, compare two strings, and concatenate two strings.

- Explored options to use standard library functions (`strlen`, `strcpy`, `strcmp`, `strcat`) and manually implement the string operations using loops.

- After user instructions, replaced standard library functions with manual implementations to copy, compare, and concatenate strings.

2. Program logic and methods used:

- The program uses a struct named `String` to represent strings in the ADT. It contains a character array `data` to hold the string data and an integer `length` to store the length of the string.

- Functions like `create_string`, `create_string_from_cstr`, `copy_string`, `compare_strings`, and `concatenate_strings` perform respective string operations using loops instead of using standard library functions.

- `MAX_STRING_LENGTH` is used as a defined constant to limit the maximum length of strings.

- The `concatenate_strings` function has basic error handling to prevent buffer overflow.

3. Importance of the approach:

- The chosen approach demonstrates the implementation of a basic Abstract Data Type (ADT) for strings using a struct. It encapsulates the implementation details within functions, improving usability and maintainability.

- Manual implementation of string operations provides a better understanding of the lower-level workings, improving comprehension of string handling using loops and character arrays.

- Using a fixed-size buffer (`data` array) with a defined maximum length (`MAX_STRING_LENGTH`) enhances program safety by preventing buffer overflows.

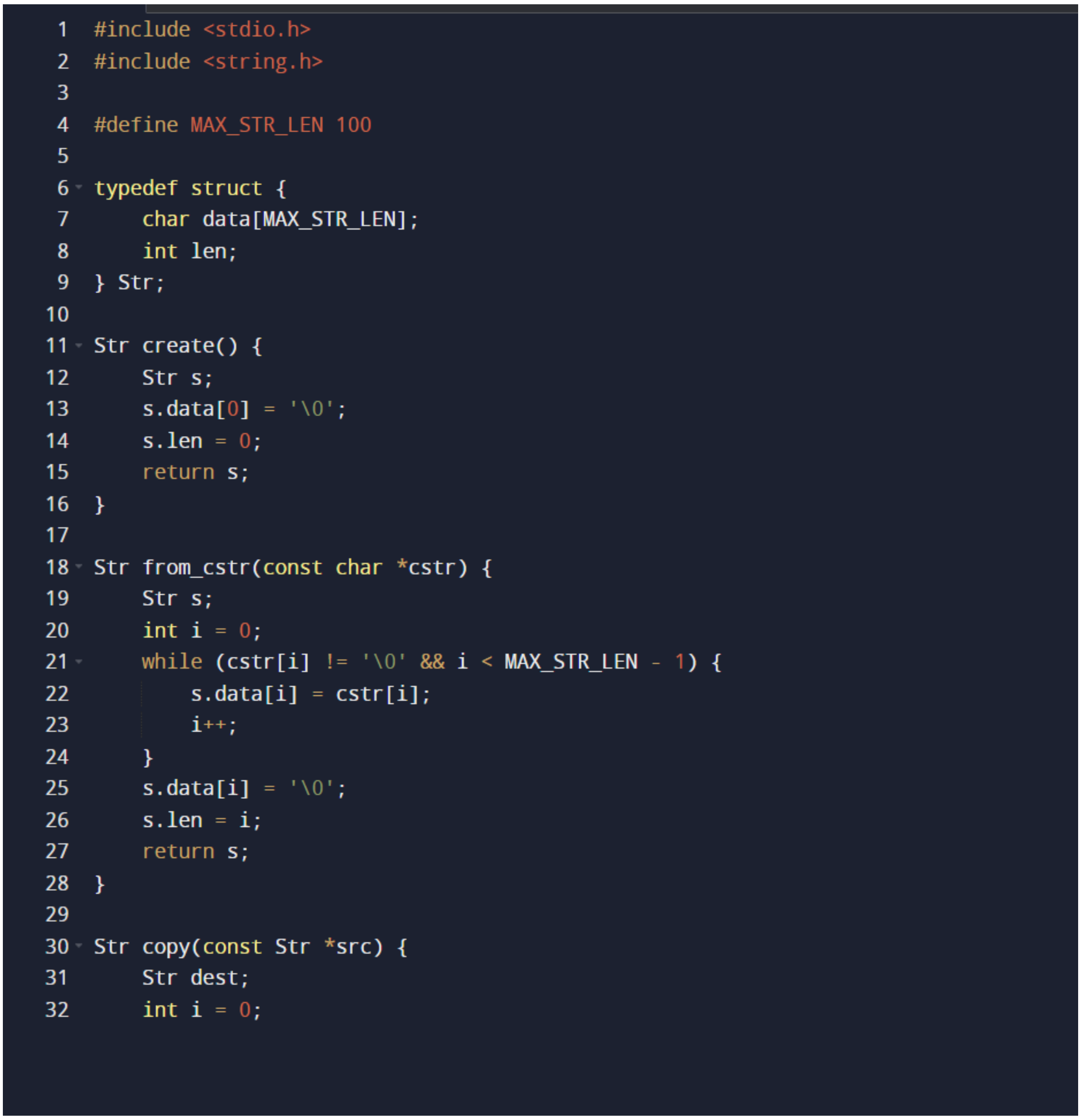
- The approach is flexible, allowing adjustment of the maximum string length based on specific requirements or system limitations.

- Employing an ADT facilitates code reuse and extension to support additional string operations or features in larger programs.

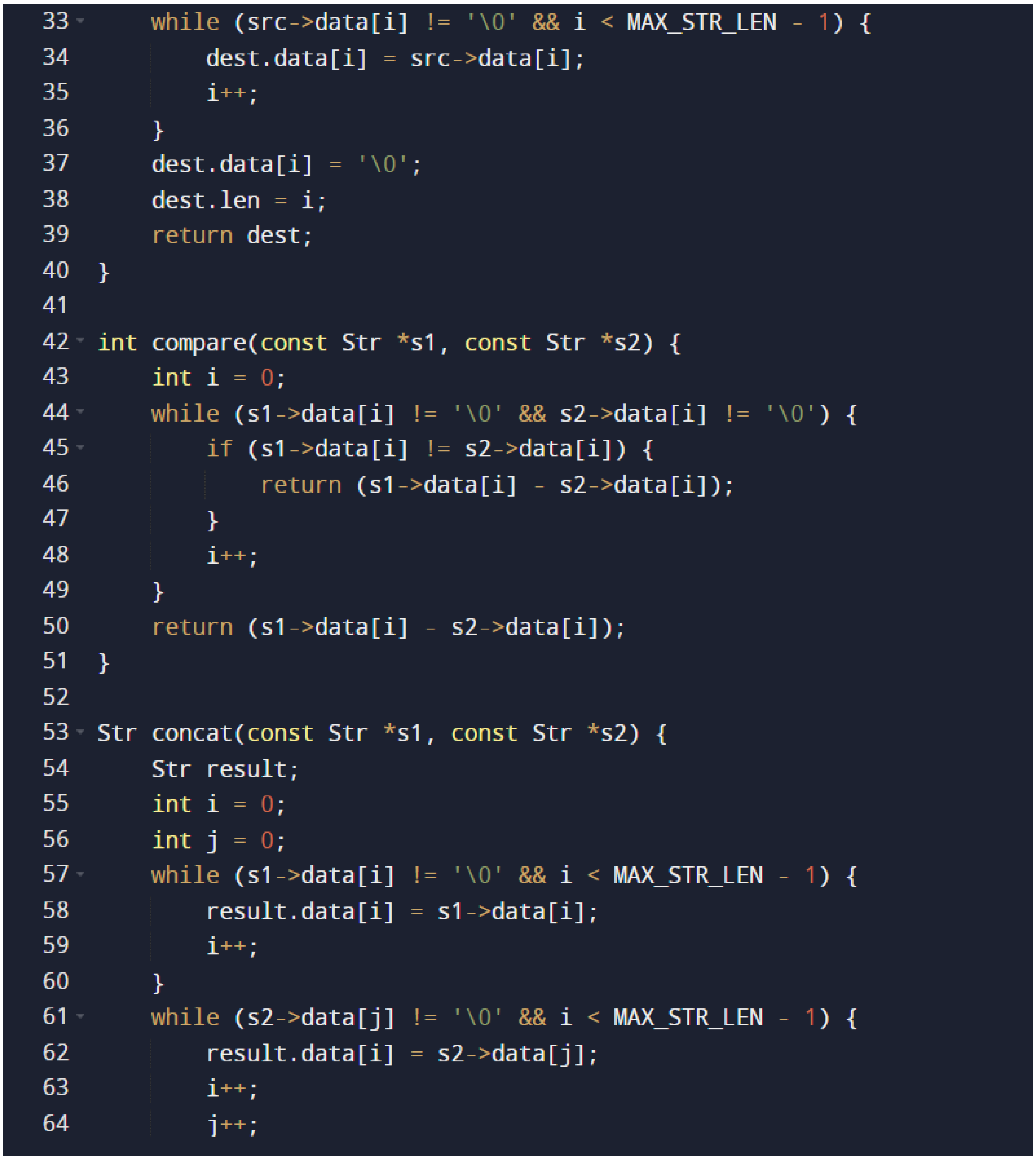
- Overall, the approach showcases encapsulation, modularity, and memory management principles in a simple yet effective string ADT implementation in C.

Program code and Output screenshots:

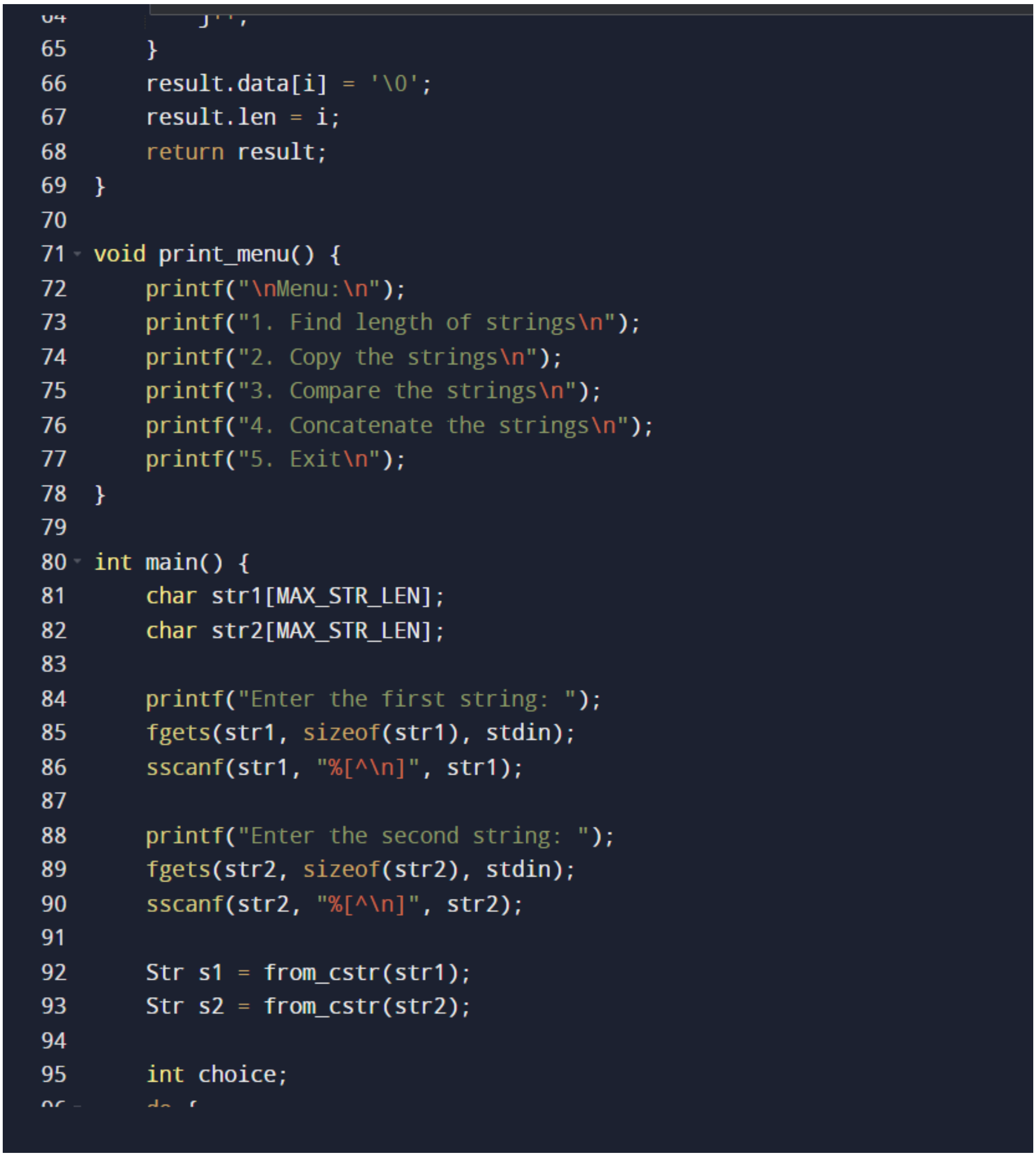
code :



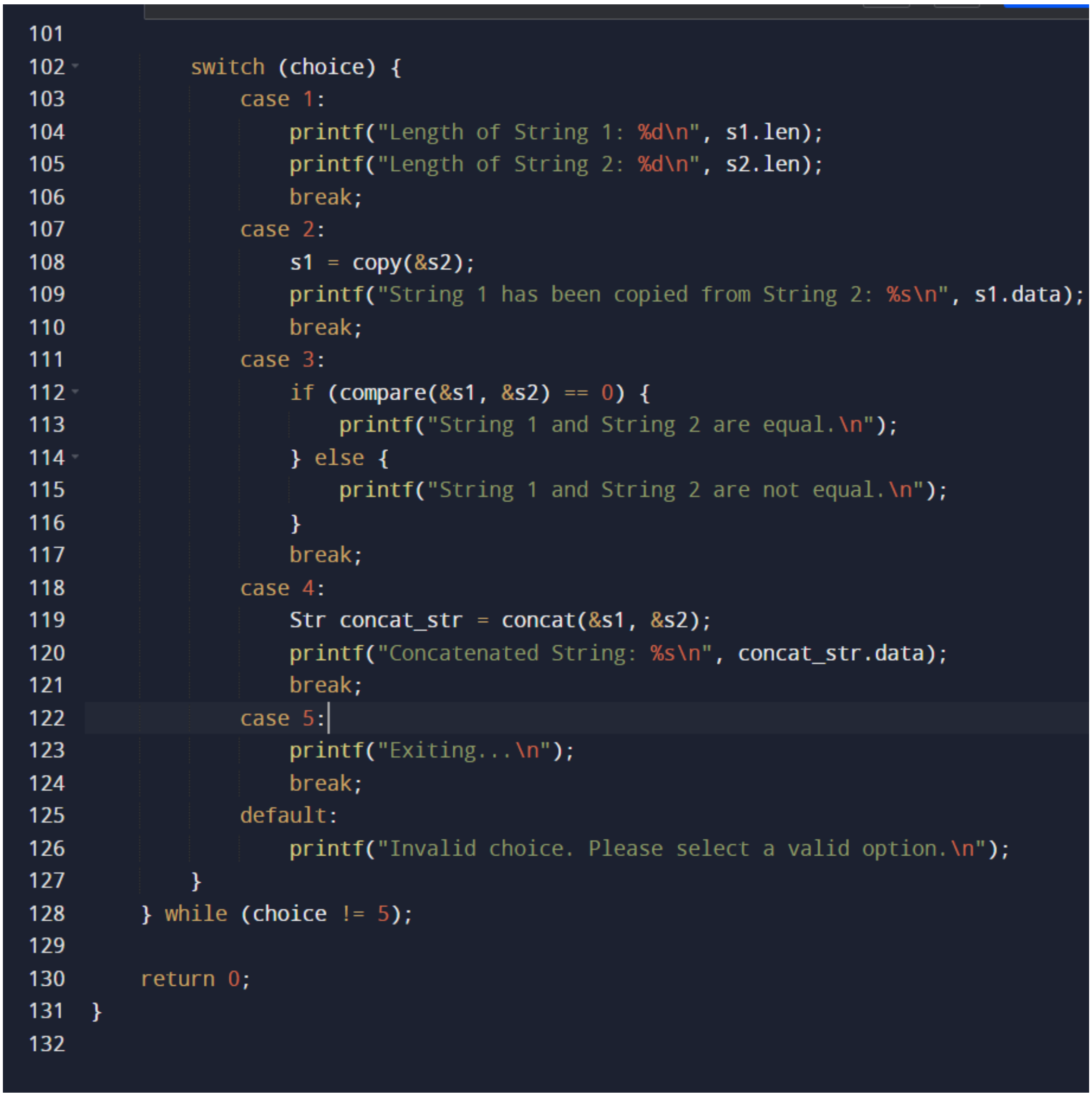
```c
#include <stdio.h>
#include <string.h>

#define MAX_STR_LEN 100

typedef struct {
    char data[MAX_STR_LEN];
    int len;
} Str;

Str create() {
    Str s;
    s.data[0] = '\0';
    s.len = 0;
    return s;
}

Str from_cstr(const char *cstr) {
    Str s;
    int i = 0;
    while (cstr[i] != '\0' && i < MAX_STR_LEN - 1) {
        s.data[i] = cstr[i];
        i++;
    }
    s.data[i] = '\0';
    s.len = i;
    return s;
}

Str copy(const Str *src) {
    Str dest;
    int i = 0;
```



```c
    while (src->data[i] != '\0' && i < MAX_STR_LEN - 1) {
        dest.data[i] = src->data[i];
        i++;
    }
    dest.data[i] = '\0';
    dest.len = i;
    return dest;
}

int compare(const Str *s1, const Str *s2) {
    int i = 0;
    while (s1->data[i] != '\0' && s2->data[i] != '\0') {
        if (s1->data[i] != s2->data[i]) {
            return (s1->data[i] - s2->data[i]);
        }
        i++;
    }
    return (s1->data[i] - s2->data[i]);
}

Str concat(const Str *s1, const Str *s2) {
    Str result;
    int i = 0;
    int j = 0;
    while (s1->data[i] != '\0' && i < MAX_STR_LEN - 1) {
        result.data[i] = s1->data[i];
        i++;
    }
    while (s2->data[j] != '\0' && i < MAX_STR_LEN - 1) {
        result.data[i] = s2->data[j];
        i++;
        j++;
```

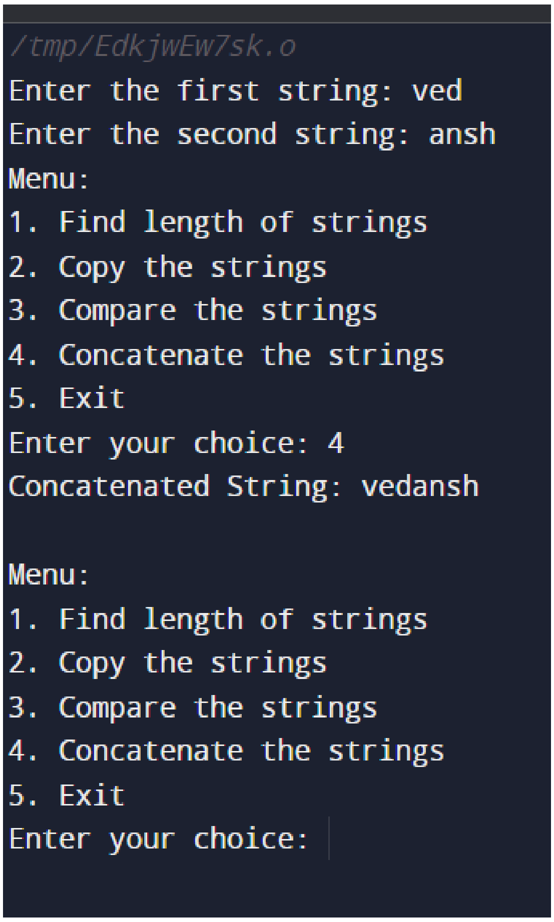


```
        j++;
    }
    result.data[i] = '\0';
    result.len = i;
    return result;
}

void print_menu() {
    printf("\nMenu:\n");
    printf("1. Find length of strings\n");
    printf("2. Copy the strings\n");
    printf("3. Compare the strings\n");
    printf("4. Concatenate the strings\n");
    printf("5. Exit\n");
}

int main() {
    char str1[MAX_STR_LEN];
    char str2[MAX_STR_LEN];

    printf("Enter the first string: ");
    fgets(str1, sizeof(str1), stdin);
    sscanf(str1, "%[^\n]", str1);

    printf("Enter the second string: ");
    fgets(str2, sizeof(str2), stdin);
    sscanf(str2, "%[^\n]", str2);

    Str s1 = from_cstr(str1);
    Str s2 = from_cstr(str2);

    int choice;
    do {
```



```

        switch (choice) {
            case 1:
                printf("Length of String 1: %d\n", s1.len);
                printf("Length of String 2: %d\n", s2.len);
                break;
            case 2:
                s1 = copy(&s2);
                printf("String 1 has been copied from String 2: %s\n", s1.data);
                break;
            case 3:
                if (compare(&s1, &s2) == 0) {
                    printf("String 1 and String 2 are equal.\n");
                } else {
                    printf("String 1 and String 2 are not equal.\n");
                }
                break;
            case 4:
                Str concat_str = concat(&s1, &s2);
                printf("Concatenated String: %s\n", concat_str.data);
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please select a valid option.\n");
        }
    } while (choice != 5);

    return 0;
}

```

output:



```
/tmp/EdkjwEw7sk.o
Enter the first string: ved
Enter the second string: ansh
Menu:
1. Find length of strings
2. Copy the strings
3. Compare the strings
4. Concatenate the strings
5. Exit
Enter your choice: 4
Concatenated String: vedansh

Menu:
1. Find length of strings
2. Copy the strings
3. Compare the strings
4. Concatenate the strings
5. Exit
Enter your choice:
```

Conclusion:-