Data Structures and algorithms

Submitted by:

Aryaveer Singh

102304064

3D12

Assignment 3

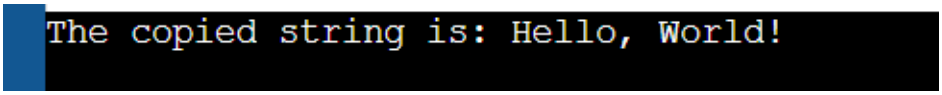
Q 1: Write a program to implement strlen() function.

```
#include <stdio.h>
int my_strlen(const char *str) {
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    return length;
}
int main() {
    char str[] = "Hello, World!";
    int len = my_strlen(str);
    printf("The length of the string is: %d\n", len);
    return 0;
}
```

```
Output                    Time: 0.003ms                    Memory: 2.83 Mb
The length of the string is: 13
```

Q 2: Write a program to implement strcpy() function.

```
#include <stdio.h>
char* my_strcpy(char *dest, const char *src) {
    int i = 0;
    while (src[i] != '\0') {
        dest[i] = src[i];
        i++;
    }
    dest[i] = '\0';
    return dest;
}
int main() {
    char source[] = "Hello, World!";
    char destination[20];
    my_strcpy(destination, source);
    printf("The copied string is: %s\n", destination);
    return 0;
}
```



Q 3: Write a program to implement strcat() function.

```
#include <stdio.h>

char* my_strcat(char *dest, const char *src) {
    int i = 0;
    int j = 0;
    while (dest[i] != '\0') {
        i++;
    }
```

```
    while (src[j] != '\0') {
        dest[i] = src[j];
        i++;
        j++;
    }
    dest[i] = '\0';
    return dest;
}

int main() {
    char str1[50] = "Hello, ";
    char str2[] = "World!";
    my_strcat(str1, str2);
    printf("The concatenated string is: %s\n", str1);
    return 0;
}
```

```
The concatenated string is: Hello, World!
```

Q 4: Write a program to implement strcmp() function.

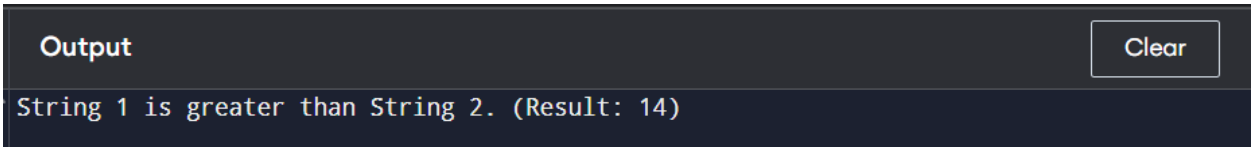
```
#include <stdio.h>

int my_strcmp(const char *str1, const char *str2) {
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0') {
        if (str1[i] != str2[i]) {
            return str1[i] - str2[i];
        }
        i++;
```

```
    }
    return str1[i] - str2[i];
}

int main() {
    char str1[] = "hello";
    char str2[] = "hella";
    int result = my_strcmp(str1, str2);

    if (result == 0) {
        printf("Strings are equal.\n");
    } else if (result > 0) {
        printf("String 1 is greater than String 2.\n");
    } else {
        printf("String 1 is less than String 2.\n");
    }
    return 0;
}
```



Q 5: WAP to demonstrate limitations of Two-Dimensional Array of Characters.

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

int main() {
    char names[3][20] = {"John", "Jane", "Alice"};
```

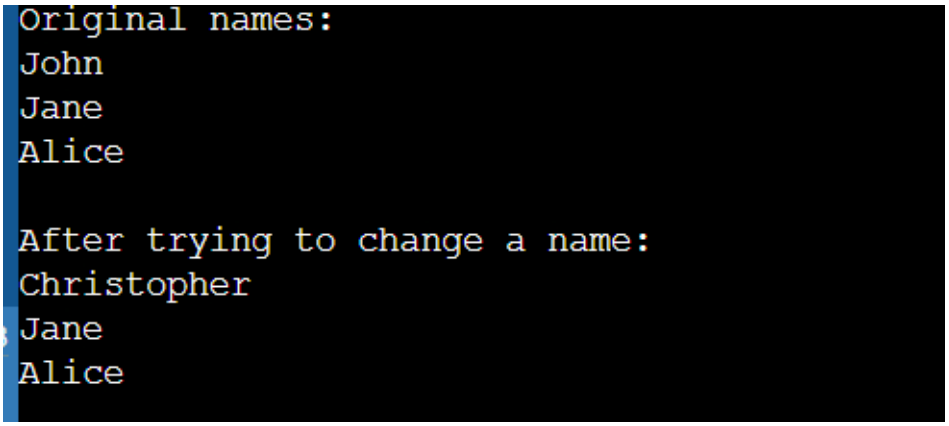
```
    Limitation 1: Fixed size for each string
    If a name is longer than 19 characters (plus null terminator), it will cause a buffer overflow.
    E.g., strcpy(names[0], "Christopher");

   printf("Original names:\n");
   for (int i = 0; i < 3; i++) {
      printf("%s\n", names[i]);
   }

    Limitation 2: Cannot change string size at runtime
    You cannot change the length of a string stored in the 2D array.
    The following would be an error:
    names[0] = "Christopher";

    You can only copy into the fixed-size buffer
   char newName[] = "Christopher";
   if (strlen(newName) < 20) {
      strcpy(names[0], newName);
   } else {
      printf("Cannot copy. New name is too long.\n");
   }
   printf("\nAfter trying to change a name:\n");
   for (int i = 0; i < 3; i++) {
      printf("%s\n", names[i]);
   }

   return 0;
}
```



Q 6: WAP to demonstrate an array of Pointers to Strings.

```
#include <stdio.h>

int main() {
   char *names[] = {"John", "Christopher", "Jane", "Alice"};

    Advantage: Pointers can point to strings of different lengths.
    This saves memory compared to a 2D array where each row has a fixed size.

   printf("Names using an array of pointers:\n");
   for (int i = 0; i < 4; i++) {
      printf("%s\n", names[i]);
   }

    Changing a pointer to point to a different string
   names[1] = "Peter";
   printf("\nAfter changing a pointer:\n");
   for (int i = 0; i < 4; i++) {
      printf("%s\n", names[i]);
```

```
    }

    return 0;
}
```

```
Names using an array of pointers:
John
Christopher
Jane
Alice

After changing a pointer:
John
Peter
Jane
Alice
```