Rajalakshmi Engineering College

Name: Manju Parkavi R
Email: 240801193@rajalakshmi.edu.in
Roll no: 2116240801193
Phone: 7397317293
Branch: REC
Department: I ECE FB
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Milton is a diligent clerk at a school who has been assigned the task of managing class schedules. The school has various sections, and Milton needs to keep track of the class schedules for each section using a stack-based system.

He uses a program that allows him to push, pop, and display class schedules for each section. Milton's program uses a stack data structure, and each class schedule is represented as a character. Help him write a program using a linked list.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the class schedule to be pushed onto the stack.

Choice 2: Pop class schedule from the stack

Choice 3: Display the class schedules in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

- If the choice is 1, push the given class schedule to the stack and display the following: "Adding Section: [class schedule]"
- If the choice is 2, pop the class schedule from the stack and display the following: "Removing Section: [class schedule]"
- If the choice is 2, and if the stack is empty without any class schedules, print "Stack is empty. Cannot pop."
- If the choice is 3, print the class schedules in the stack in the following: "Enrolled Sections: " followed by the class schedules separated by space.
- If the choice is 3, and there are no class schedules in the stack, print "Stack is empty"
- If the choice is 4, exit the program and display the following: "Exiting the program"
- If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact format.

Sample Test Case

Input: 1 d
1 h
3
2

```
3
4
Output: Adding Section: d
Adding Section: h
Enrolled Sections: h d
Removing Section: h
Enrolled Sections: d
Exiting program
```

Answer

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

struct Node {
    char data;
    struct Node* next;
};

struct Node* top = NULL;

// You are using GCC
void push(char value) {
    // Create a new node
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        // Handle memory allocation failure (acts like overflow in fixed-size stacks)
        printf("Memory allocation failed. Cannot add section.\n"); // Matches
previous output for this case
        return;
    }

    // Assign the class schedule character to the new node
    newNode->data = value;

    // Link the new node to the current top
    newNode->next = top;

    // Update the top of the stack to the new node
    top = newNode;

    // Print the push confirmation message as per the output format
    printf("Adding Section: %c\n", value);
```

```
}
// Function to pop a class schedule (character) from the stack
void pop() {
   // Check if the stack is empty (underflow condition)
   if (top == NULL) {
      printf("Stack is empty. Cannot pop.\n"); // Output format for empty pop
(matches sample with period)
      return;
   }

   // Store the top node to free it later
   struct Node* temp = top;

   // Get the class schedule character before popping
   char poppedValue = temp->data;

   // Update the top to the next node
   top = top->next;

   // Free the memory of the popped node
   free(temp);

   // Print the pop confirmation message as per the output format
   printf("Removing Section: %c\n", poppedValue); // Matches output format
}

// Function to display the class schedules in the stack
void displayStack() {
   // Check if the stack is empty
   if (top == NULL) {
      printf("Stack is empty\n"); // Output format for empty display
      return;
   }

   // Traverse the stack from top to bottom and print elements
   struct Node* current = top;
   printf("Enrolled Sections: "); // Output format for non-empty display
   while (current != NULL) {
      printf("%c ", current->data); // Print character followed by a space
      current = current->next;
   }
```

```
    printf("\n"); // Newline after displaying elements
}
int main() {
    int choice;
    char value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
                printf("Exiting program\n");
                break;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

Status : Correct ***Marks : 10/10***