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 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sanjeev is in charge of managing a library's book storage, and he wants to create a program that simplifies this task. His goal is to implement a program that simulates a stack using an array.

Help him in writing a program that provides the following functionality:

Add Book ID to the Stack (Push): You can add a book ID to the top of the book stack. Remove Book ID from the Stack (Pop): You can remove the top book ID from the stack and display its details. If the stack is empty, you cannot remove any more book IDs.Display Books ID in the Stack (Display): You can view the books ID currently on the stack.Exit the Library: You can choose to exit the program.

Input Format

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

Choice 1: Push the book onto the stack. If the choice is 1, the following input is a space-separated integer, representing the ID of the book to be pushed onto the stack.

Choice 2: Pop the book ID from the stack.

Choice 3: Display the book ID in the stack.

Choice 4: Exit the program.

Output Format

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

1. If the choice is 1, push the given book ID to the stack and display the corresponding message.
2. If the choice is 2, pop the book ID from the stack and display the corresponding message.
3. If the choice is 2, and if the stack is empty without any book ID, print "Stack Underflow"
4. If the choice is 3, print the book IDs in the stack.
5. If the choice is 3, and there are book IDs in the stack, print "Stack is empty"
6. If the choice is 4, exit the program and display the corresponding message.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1 19
1 28
2
3
2
4
Output: Book ID 19 is pushed onto the stack
Book ID 28 is pushed onto the stack

Book ID 28 is popped from the stack
Book ID in the stack: 19
Book ID 19 is popped from the stack
Exiting the program

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a stack node (representing a book ID)
struct Node {
    int data;
    struct Node* next;
};

// Global variable for the top of the stack
struct Node* top = NULL;

// Function to push a book ID onto the stack
void push(int value) {
    // Create a new node
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        // Handle memory allocation failure (overflow in practical terms for this
scenario)
        // The sample output doesn't show an overflow case, so we'll just print a
message.
        printf("Memory allocation failed. Cannot push element.\n");
        return;
    }

    // Assign book ID to the new node
    newNode->data = value;

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

    // Update the top of the stack
    top = newNode;

    // Print the push confirmation message as per the sample output format
```

```
    printf("Book ID %d is pushed onto the stack\n", value);
}

// Function to pop a book ID from the stack
void pop() {
    // Check if the stack is empty (underflow condition)
    if (top == NULL) {
        printf("Stack Underflow\n"); // Output format for empty pop (matches
sample)
        return;
    }

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

    // Get the book ID before popping
    int 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 sample output format
    printf("Book ID %d is popped from the stack\n", poppedValue);
}

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

    // Traverse the stack from top to bottom and print elements
    struct Node* current = top;
    printf("Book ID in the stack: "); // Output format for non-empty display
(matches sample)
    while (current != NULL) {
```

```
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n"); // Newline after displaying elements
}

int main() {
    int choice;
    int bookID;

    // Loop to keep the program running until the user exits
    while (1) {
        // No menu printing here to match the sample input/output format which
doesn't show prompts
        // You would typically print a menu in a real interactive program.

        // Read the user's choice
        if (scanf("%d", &choice) != 1) {
            // Handle potential non-integer input gracefully, though not strictly
required by sample
            printf("Invalid input. Please enter a number.\n");
            while (getchar() != '\n'); // Consume the invalid input
            continue; // Go to the next loop iteration
        }


        // Process the user's choice
        switch (choice) {
            case 1:
                // For choice 1, we also need to read the book ID
                if (scanf("%d", &bookID) != 1) {
                    printf("Invalid input for Book ID.\n");
                    while (getchar() != '\n'); // Consume the invalid input
                    // Decide whether to continue or break based on desired error
handling
                    continue; // Skip the push operation for this invalid input
                }
                push(bookID);
                break;
            case 2:
                pop();
                break;
```

```
            case 3:
                displayStack();
                break;
            case 4:
                printf("Exiting the program\n"); // Output format for exiting (matches
sample)
                // Before exiting, it's good practice to free all allocated memory
                while (top != NULL) {
                    struct Node *temp = top;
                    top = top->next;
                    free(temp);
                }
                return 0; // Exit the program
            default:
                printf("Invalid choice\n"); // Output format for invalid choice (matches
sample)
                // Consume the rest of the line to prevent issues with future inputs
                // Note: For single number invalid choice like '9' followed by '4', this is
sufficient.
                // If the invalid input was '9 abc', the initial scanf might leave ' abc' in the
buffer.
                // For competitive programming like scenarios where input format is
guaranteed, simpler handling might be enough.
                // Let's assume the sample input format implies valid integers for
choices and book IDs when expected.
                break; // Just print invalid choice and wait for the next input
        }
    }

return 0;
}
```

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