Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a coding competition, you are assigned a task to create a program that simulates a stack using a linked list.

The program should feature a menu-driven interface for pushing an integer to stack, popping, and displaying stack elements, with robust error handling for stack underflow situations. This challenge tests your data structure skills.

Input Format

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

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

the stack.

Choice 2: Pop the integer from the stack.

Choice 3: Display the elements 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 integer to the stack and display the following: "Pushed element: " followed by the value pushed.

If the choice is 2, pop the integer from the stack and display the following: "Popped element: " followed by the value popped.

If the choice is 2, and if the stack is empty without any elements, print "Stack is empty. Cannot pop."

If the choice is 3, print the elements in the stack: "Stack elements (top to bottom): " followed by the space-separated values.

If the choice is 3, and there are no elements in the stack, print "Stack is empty".

If the choice is 4, exit the program and display the following: "Exiting program".

241901124

If any other choice is entered, print "Invalid choice".

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

Sample Test Case

Input: 1 3
1 4
3
2
3
4
Output: Pushed element: 3
Pushed element: 4
Stack elements (top to bottom): 4 3
Popped element: 4
Stack elements (top to bottom): 3
Exiting program

Answer

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

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

struct Node* top = NULL;

void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Heap overflow\n");
        return;
    }
    newNode->data = value;
    newNode->next = top;
    top = newNode;
```

```
    printf("Pushed element: %d\n", value);
}

void pop() {
    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
        return;
    }
    struct Node* temp = top;
    printf("Popped element: %d\n", top->data);
    top = top->next;
    free(temp);
}

void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct Node* temp = top;
    printf("Stack elements (top to bottom): ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int choice, value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
```

241901124

```
            case 4:
                printf("Exiting program\n");
                return 0;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

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

241901124

Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

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

```
#include <stdio.h>
#define MAX 100

int stack[MAX];
int top = -1;

void push(int bookID) {
    if (top >= MAX - 1) {
        // Optional: Add overflow handling if required.
        return;
    }
    top++;
    stack[top] = bookID;
    printf("Book ID %d is pushed onto the stack\n", bookID);
}

void pop() {
    if (top == -1) {
        printf("Stack Underflow\n");
    } else {
        printf("Book ID %d is popped from the stack\n", stack[top]);
        top--;
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Book ID in the stack: ");
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
```

```
int main() {
    int choice, bookID;

    while (1) {
        if (scanf("%d", &choice) != 1) {
            break; // Exit on invalid input
        }

        switch (choice) {
            case 1:
                if (scanf("%d", &bookID) != 1) {
                    // Invalid push input
                    while (getchar() != '\n'); // clear the buffer
                    continue;
                }
                push(bookID);
                break;

            case 2:
                pop();
                break;

            case 3:
                display();
                break;

            case 4:
                printf("Exiting the program\n");
                return 0;

            default:
                printf("Invalid choice\n");
                break;
        }
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sharon is developing a programming challenge for a coding competition. The challenge revolves around implementing a character-based stack data structure using an array.

Sharon's project involves a stack that can perform the following operations:

Push a Character: Users can push a character onto the stack.Pop a Character: Users can pop a character from the stack, removing and displaying the top character.Display Stack: Users can view the current elements in the stack.Exit: Users can exit the stack operations application.

Write a program to help Sharon to implement a program that performs the given operations.

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 character to be pushed onto the stack.

Choice 2: Pop the character from the stack.

Choice 3: Display the characters 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 character to the stack and display the pushed character having the prefix "Pushed: ".
2. If the choice is 2, undo the character from the stack and display the character that is popped having the prefix "Popped: ".
3. If the choice is 2, and if the stack is empty without any characters, print "Stack is empty. Nothing to pop."
4. If the choice is 3, print the elements in the stack having the prefix "Stack elements: ".
5. If the choice is 3, and there are no characters in the stack, print "Stack is empty."
6. If the choice is 4, exit the program.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
4

Output: Stack is empty. Nothing to pop.

Answer

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

```
#include <stdbool.h>

#define MAX_SIZE 100

char items[MAX_SIZE];
int top = -1;

void initialize() {
   top = -1;
}
bool isFull() {
   return top == MAX_SIZE - 1;
}

bool isEmpty() {
   return top == -1;
}
void push(char ch) {
   if (top >= MAX_SIZE - 1) {
      // Optional: handle stack overflow
      return;
   }
   items[++top] = ch;
   printf("Pushed: %c\n", ch);
}

void pop() {
   if (top == -1) {
      printf("Stack is empty. Nothing to pop.\n");
   } else {
      printf("Popped: %c\n", items[top--]);
   }
}

void display() {
   if (top == -1) {
      printf("Stack is empty.\n");
   } else {
      printf("Stack elements: ");
      for (int i = top; i >= 0; i--) {
         printf("%c ", items[i]);
      }
```

```
        printf("\n");
    }
}

int main() {
    initialize();
    int choice;
    char value;

    while (true) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

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

241901124

Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are a software developer tasked with building a module for a scientific calculator application. The primary function of this module is to convert infix mathematical expressions, which are easier for users to read and write, into postfix notation (also known as Reverse Polish Notation). Postfix notation is more straightforward for the application to evaluate because it removes the need for parentheses and operator precedence rules.

The scientific calculator needs to handle various mathematical expressions with different operators and ensure the conversion is correct. Your task is to implement this infix-to-postfix conversion algorithm using a stack-based approach.

Example

Input:

a+b

Output:

ab+

Explanation:

The postfix representation of (a+b) is ab+.

Input Format

The input is a string, representing the infix expression.

Output Format

The output displays the postfix representation of the given infix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a+(b*e)

Output: abe*+

Answer

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

struct Stack {
    int top;
    unsigned capacity;
    char* array;
};

struct Stack* createStack(unsigned capacity) {
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));

    if (!stack)
```

```
        return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (char*)malloc(stack->capacity * sizeof(char));

    return stack;
}

int isEmpty(struct Stack* stack) {
    return stack->top == -1;
}

char peek(struct Stack* stack) {
    return stack->array[stack->top];
}

char pop(struct Stack* stack) {
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack* stack, char op) {
    stack->array[++stack->top] = op;
}

int isOperand(char ch) {
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9');
}

int precedence(char ch) {
    switch (ch) {
        case '^': return 3;
        case '*':
        case '/': return 2;
        case '+':
        case '-': return 1;
        default: return 0;
    }
}

int isRightAssociative(char ch) {
```

```
    return ch == '^';  // '^' is right associative
}

void infixToPostfix(char* exp) {
    struct Stack* stack = createStack(strlen(exp)); // Create a stack to hold
operators
    char output[100];  // Array to store the postfix expression
    int k = 0;  // Index for the output

    for (int i = 0; exp[i]; i++) {
        char ch = exp[i];

        if (isOperand(ch)) {  // If it's an operand (either a number or a letter)
            output[k++] = ch;
        } else if (ch == '(') {  // If it's an opening parenthesis, push to stack
            push(stack, ch);
        } else if (ch == ')') {  // If it's a closing parenthesis, pop until '('
            while (!isEmpty(stack) && peek(stack) != '(') {
                output[k++] = pop(stack);
            }
            pop(stack);  // Pop the '(' from the stack
        } else {  // If it's an operator
            while (!isEmpty(stack) && precedence(ch) <= precedence(peek(stack)) &&
                   !isRightAssociative(ch)) {
                output[k++] = pop(stack);
            }
            push(stack, ch);
        }
    }

    // Pop all remaining operators from the stack
    while (!isEmpty(stack)) {
        output[k++] = pop(stack);
    }

    output[k] = '\0';  // Null-terminate the postfix expression
    printf("%s\n", output);  // Print the postfix expression
}

int main() {
    char exp[100];
    scanf("%s", exp);
```

```
    infixToPostfix(exp);
    return 0;
}
```

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

241901124

Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

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;

void push(char value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  //
Allocate memory for a new node
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = value;  // Set the data for the new node
    newNode->next = top;    // Point to the current top node
    top = newNode;          // Update the top to the new node
    printf("Adding Section: %c\n", value);
}

// Pop function to remove an element from the stack
void pop() {
    if (top == NULL) {
        // If the stack is empty, we can't pop
        printf("Stack is empty. Cannot pop.\n");
        return;
    }

    struct Node* temp = top;     // Temporary pointer to hold the current top node
```

```
    top = top->next;            // Move the top pointer to the next node
    printf("Removing Section: %c\n", temp->data);  // Print the removed section
    free(temp);  // Free the memory of the popped node
}

// Display function to print all elements in the stack
void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }

    struct Node* temp = top;
    printf("Enrolled Sections: ");
    while (temp != NULL) {
        printf("%c ", temp->data);  // Print each section
        temp = temp->next;          // Move to the next node
    }
    printf("\n");
}

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**

241901124