# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

## NeoColab_REC_CS23231_DATA STRUCTURES

### REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 17

### Section 1 : MCQ

1. Elements are Added on _____ of the Stack.

***Answer***

Top

***Status :*** Correct ***Marks : 1/1***

2. What is the value of the postfix expression 6 3 2 4 + - *?

***Answer***

-18

***Status :*** Correct ***Marks : 1/1***

3. Which of the following operations allows you to examine the top element of a stack without removing it?

***Answer***

Peek

***Status :*** Correct ***Marks : 1/1***

4. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
int isEmpty() {
    return (top == -1);
}
int isFull() {
    return (top == MAX_SIZE - 1);
}
void push(int item) {
    if (isFull())
        printf("Stack Overflow\n");
    else
        stack[++top] = item;
}
int main() {
    printf("%d\n", isEmpty());
    push(10);
    push(20);
    push(30);
    printf("%d\n", isFull());
    return 0;
}
```

***Answer***

10

***Status :*** Correct ***Marks : 1/1***

5. What is the advantage of using a linked list over an array for implementing a stack?

***Answer***

Linked lists can dynamically resize

***Status :*** Correct ***Marks : 1/1***

6. Here is an Infix Expression: 4+3*(6*3-12). Convert the expression from Infix to Postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression?

***Answer***

4

***Status :*** Correct ***Marks : 1/1***

7. In a stack data structure, what is the fundamental rule that is followed for performing operations?

***Answer***

Last In First Out

***Status :*** Correct ***Marks : 1/1***

8. Pushing an element into the stack already has five elements. The stack size is 5, then the stack becomes

***Answer***

Overflow

***Status :*** Correct ***Marks : 1/1***

9. The user performs the following operations on the stack of size 5 then at the end of the last operation, the total number of elements present in the stack is

push(1);
pop();
push(2);
push(3);
pop();
push(4);
pop();
pop();
push(5);

***Answer***

1

***Status :*** Correct ***Marks : 1/1***

10. A user performs the following operations on stack of size 5 then which of the following is correct statement for Stack?

push(1);
pop();
push(2);
push(3);
pop();
push(2);
pop();
pop();
push(4);
pop();
pop();
push(5);

***Answer***

Underflow Occurs

***Status :*** Correct ***Marks : 1/1***

11. In an array-based stack, which of the following operations can result in a Stack underflow?

***Answer***

Popping an element from an empty stack

***Status :*** Correct ***Marks : 1/1***

12. When you push an element onto a linked list-based stack, where does the new element get added?

***Answer***

At the end of the list

***Status :*** Wrong ***Marks : 0/1***

13. Which of the following Applications may use a Stack?

***Answer***

All of the mentioned options

***Status :*** Correct ***Marks : 1/1***

14. Consider a linked list implementation of stack data structure with three operations:

push(value): Pushes an element value onto the stack.pop(): Pops the top element from the stack.top(): Returns the item stored at the top of the stack.

Given the following sequence of operations:

push(10);pop();push(5);top();

What will be the result of the stack after performing these operations?

***Answer***

The top element in the stack is 5

***Status :*** Correct ***Marks : 1/1***

15. What is the primary advantage of using an array-based stack with a fixed size?

***Answer***

None of the mentioned options

***Status :*** Wrong ***Marks : 0/1***

16. In the linked list implementation of the stack, which of the following operations removes an element from the top?

***Answer***

Pop

***Status :*** Correct ***Marks : 1/1***

17. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
void display() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements: ");
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
void push(int value) {
    if (top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
    } else {
        stack[++top] = value;
```

```
    }
}
int main() {
    display();
    push(10);
    push(20);
    push(30);
    display();
    push(40);
    push(50);
    push(60);
    display();
    return 0;
}
```

***Answer***

Stack is emptyStack elements: 30 20 10Stack OverflowStack elements: 50 40 30 20 10 

***Status :*** Correct ***Marks : 1/1***

18. Consider the linked list implementation of a stack.

Which of the following nodes is considered as Top of the stack?

***Answer***

Last node

***Status :*** Wrong ***Marks : 0/1***

19. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
void push(int* stack, int* top, int item) {
    if (*top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        return;
    }
```

```
    stack[++(*top)] = item;
}
int pop(int* stack, int* top) {
    if (*top == -1) {
        printf("Stack Underflow\n");
        return -1;
    }
    return stack[(*top)--];
}

int main() {
    int stack[MAX_SIZE];
    int top = -1;
    push(stack, &top, 10);
    push(stack, &top, 20);
    push(stack, &top, 30);
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    return 0;
}
```

***Answer***

302010Stack Underflow-1

***Status :*** Correct ***Marks : 1/1***

20. The result after evaluating the postfix expression 10 5 + 60 6 / * 8 - is

***Answer***

142

***Status :*** Correct ***Marks : 1/1***

# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
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".

241901035

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));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
    printf("Pushed element: %d\n", value);
}

int pop() {
```

```
    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
        return -1;
    }
    struct Node* temp = top;
    int poppedValue = temp->data;
    top = top->next;
    free(temp);
    printf("Popped element: %d\n", poppedValue);
    return poppedValue;
}

void displayStack() {
    struct Node* current = top;
    if (current == NULL) {
        printf("Stack is empty\n");
        return;
    }
    printf("Stack elements (top to bottom):");
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->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;
            case 4:
                printf("Exiting program\n");
```

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

    return 0;
}
```

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

241901035

# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
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 value) {
    if (top == MAX - 1) {
        printf("Stack Overflow\n");
    } else {
        stack[++top] = value;
        printf("Book ID %d is pushed onto the stack\n", value);
    }
}

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

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

int main() {
```

```
    int choice, value;

    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d", &value);
                push(value);
                break;

            case 2:
                pop();
                break;

            case 3:
                display();
                break;

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

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

    return 0;
}
```

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

# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
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 character) {
   if (top >= MAX_SIZE - 1) {
      printf("Stack Overflow\n");
      return;
   }
   items[++top] = character;
   printf("Pushed: %c\n", character);
}

void pop() {
   if (top < 0) {
      printf("Stack is empty. Nothing to pop.\n");
      return;
   }
   char poppedChar = items[top--];
   printf("Popped: %c\n", poppedChar);
}

void display() {
   if (top < 0) {
      printf("Stack is empty.\n");
      return;
   }
   printf("Stack elements: ");
   for (int i = top; i >= 0; i--) {
```

```
            printf("%c", items[i]);
            if (i > 0) {
                printf(" ");
            }
        }
        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**

241901035

# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
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');
}

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

void infixToPostfix(char* exp) {
```

241901035

```
    struct Stack* stack = createStack(strlen(exp));
    char* result = (char*)malloc(strlen(exp) + 1);
    int k = 0;

    for (int i = 0; exp[i]; i++) {
        if (isOperand(exp[i])) {
            result[k++] = exp[i];
        } else if (exp[i] == '(') {
            push(stack, exp[i]);
        } else if (exp[i] == ')') {
            while (!isEmpty(stack) && peek(stack) != '(') {
                result[k++] = pop(stack);
            }
            pop(stack);
        } else {
            while (!isEmpty(stack) && Prec(peek(stack)) >= Prec(exp[i])) {
                result[k++] = pop(stack);
            }
            push(stack, exp[i]);
        }
    }

    while (!isEmpty(stack)) {
        result[k++] = pop(stack);
    }

    result[k] = '\0';
    printf("%s\n", result);
    free(stack->array);
    free(stack);
    free(result);
}

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

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

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

# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
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));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
    printf("Adding Section: %c\n", value);
}

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

void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
```

```
        return;
    }
    struct Node* current = top;
    printf("Enrolled Sections: ");
    while (current != NULL) {
        printf("%c", current->data);
        current = current->next;
        if (current != NULL) {
            printf(" ");
        }
    }
    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*

# Rajalakshmi Engineering College

Name: Harish Vidyarth
Email: 241901035@rajalakshmi.edu.in
Roll no: 241901035
Phone: 6382551983
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

## NeoColab_REC_CS23231_DATA STRUCTURES

### REC_DS using C_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

### Section 1 : Coding

1. Problem Statement

Latha is taking a computer science course and has recently learned about infix and postfix expressions. She is fascinated by the idea of converting infix expressions into postfix notation. To practice this concept, she wants to implement a program that can perform the conversion for her.

Help Latha by designing a program that takes an infix expression as input and outputs its equivalent postfix notation.

Example

Input:

(3+4)5

Output:

34+5

***Input Format***

The input consists of a string, the infix expression to be converted to postfix notation.

***Output Format***

The output displays a string, the postfix expression equivalent of the input infix expression.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: A+B*C-D/E

Output: ABC*+DE/-

***Answer***

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

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

struct Stack* createStack(unsigned capacity) {
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    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 isalnum(ch);
}

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

void infixToPostfix(char* exp, char* result) {
    struct Stack* stack = createStack(strlen(exp));
    int k = 0;

    for (int i = 0; exp[i]; i++) {
        if (isOperand(exp[i])) {
            result[k++] = exp[i];
        } else if (exp[i] == '(') {
            push(stack, exp[i]);
        } else if (exp[i] == ')') {
            while (!isEmpty(stack) && peek(stack) != '(') {
```

```
            result[k++] = pop(stack);
        }
        pop(stack);
    } else {
        while (!isEmpty(stack) && precedence(peek(stack)) >=
precedence(exp[i])) {
            result[k++] = pop(stack);
        }
        push(stack, exp[i]);
    }
  }

  while (!isEmpty(stack)) {
    result[k++] = pop(stack);
  }

  result[k] = '\0';
  free(stack->array);
  free(stack);
}

int main() {
  char exp[101];
  char result[101];

  scanf("%100s", exp);
  infixToPostfix(exp, result);
  printf("%s\n", result);

  return 0;
}
```

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

2. Problem Statement

Rithi is building a simple text editor that allows users to type characters, undo their typing, and view the current text. She has implemented this text editor using an array-based stack data structure.

She has to develop a basic text editor with the following features:

Type a Character (Push): Users can type a character and add it to the text editor.Undo Typing (Pop): Users can undo their typing by removing the last character they entered from the editor.View Current Text (Display): Users can view the current text in the editor, which is the sequence of characters in the buffer.Exit: Users can exit the text editor application.

Write a program that simulates this text editor's undo feature using a character stack and implements the push, pop and display operations accordingly.

***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, print: "Typed character: <character>" where <character> is the character that was pushed to the stack.
2. If the choice is 2, print: "Undo: Removed character <character>" where <character> is the character that was removed from the stack.
3. If the choice is 2, and if the stack is empty without any characters, print "Text editor buffer is empty. Nothing to undo."
4. If the choice is 3, print: "Current text: <character1> <character2> ... <characterN>" where <character1>, <character2>, ... are the characters in the stack, starting from the last pushed character.
5. If the choice is 3, and there are no characters in the stack, print "Text editor buffer 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: 1 H
1 A
3
4

Output: Typed character: H
Typed character: A
Current text: A H

***Answer***

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

#define MAX_SIZE 100

struct Stack {
    char characters[MAX_SIZE];
    int top;
};

void initStack(struct Stack* stack) {
    stack->top = -1;
}

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

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

void push(struct Stack* stack, char ch) {
    if (!isFull(stack)) {
        stack->characters[++stack->top] = ch;
```

```
    }
}

char pop(struct Stack* stack) {
    if (!isEmpty(stack)) {
        return stack->characters[stack->top--];
    }
    return '\0';
}

void display(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Text editor buffer is empty.\n");
    } else {
        printf("Current text: ");
        for (int i = stack->top; i >= 0; i--) {
            printf("%c", stack->characters[i]);
            if (i != 0) {
                printf(" ");
            }
        }
        printf("\n");
    }
}

int main() {
    struct Stack stack;
    initStack(&stack);
    int choice;
    char ch;

    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &ch);
                push(&stack, ch);
                printf("Typed character: %c\n", ch);
                break;
            case 2:
                if (isEmpty(&stack)) {
                    printf("Text editor buffer is empty. Nothing to undo.\n");
```

2419035

```
            } else {
                char removedChar = pop(&stack);
                printf("Undo: Removed character %c\n", removedChar);
            }
            break;
        case 3:
            display(&stack);
            break;
        case 4:
            return 0;
        default:
            printf("Invalid choice\n");
            break;
    }
  }

  return 0;
}
```

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

3. Problem Statement

Suppose you are building a calculator application that allows users to enter mathematical expressions in infix notation. One of the key features of your calculator is the ability to convert the entered expression to postfix notation using a Stack data structure.

Write a function to convert infix notation to postfix notation using a Stack.

***Input Format***

The input consists of a string, an infix expression that includes only digits(0-9), and operators(+, -, *, /).

***Output Format***

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

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1+2*3/4-5

Output: 123*4/+5-

***Answer***

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

#define MAX_SIZE 30

struct Stack {
    char operators[MAX_SIZE];
    int top;
};

void initStack(struct Stack* stack) {
    stack->top = -1;
}

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

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

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

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

```
    return '\0';
}

char peek(struct Stack* stack) {
    if (!isEmpty(stack)) {
        return stack->operators[stack->top];
    }
    return '\0';
}

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

void infixToPostfix(const char* infix, char* postfix) {
    struct Stack stack;
    initStack(&stack);
    int j = 0;

    for (int i = 0; infix[i] != '\0'; i++) {
        if (isdigit(infix[i])) {
            while (isdigit(infix[i])) {
                postfix[j++] = infix[i++];
            }
            postfix[j++] = ' ';
            i--;
        } else if (infix[i] == '(') {
            push(&stack, infix[i]);
        } else if (infix[i] == ')') {
            while (!isEmpty(&stack) && peek(&stack) != '(') {
                postfix[j++] = pop(&stack);
                postfix[j++] = ' ';
            }
            pop(&stack);
        } else {
            while (!isEmpty(&stack) && precedence(peek(&stack)) >=
```

```
precedence(infix[i])) {
            postfix[j++] = pop(&stack);
            postfix[j++] = ' ';
        }
        push(&stack, infix[i]);
    }
  }

  while (!isEmpty(&stack)) {
    postfix[j++] = pop(&stack);
    postfix[j++] = ' ';
  }

  postfix[j - 1] = '\0';
}

int main() {
  char infix[MAX_SIZE + 1];
  char postfix[MAX_SIZE * 2];

  scanf("%30s", infix);
  infixToPostfix(infix, postfix);

  int len = strlen(postfix);
  if (len > 0 && postfix[len - 1] == ' ') {
    postfix[len - 1] = '\0';
  }

  printf("%s\n", postfix);
  return 0;
}
```

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

241901035