Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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 : 20

Section 1 : MCQ

1. 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***

2. 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***

3. 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***

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

Answer

Efficient memory usage

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

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

Answer

At the beginning of the list

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

6. 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**

7. 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***

8. 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***

9. 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***

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

Answer

-18

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

12. Elements are Added on _____ of the Stack.

Answer

Top

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

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

Answer

All of the mentioned options

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

14. 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***

15. 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***

16. 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***

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

Answer

142

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

18. 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***

19. Consider the linked list implementation of a stack.

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

Answer

First node

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

20. 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***

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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".

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;

int IsEmpty()
{
    if(top==NULL)
    {
        return 1;
    }
    else
        return 0;
}
```

```
void push(int value){
   Node* newNode=(Node*)malloc(sizeof(Node));
   newNode->data=value;
   if(IsEmpty())
      newNode->next=NULL;
   else
      newNode->next=top;
   top=newNode;
   printf("Pushed element:%d\n",top->data);
}
void pop() {
   if(IsEmpty())
      printf("Stack is empty. Cannot pop.\n");
   else{
      Node *temp;
      temp=top;
      top=top->next;
      printf("Popped element: %d\n",temp->data);
      free(temp);
   }
}
void displayStack(){
   if(IsEmpty())
      printf("Stack is empty\n");
   else{
      Node* pos;
      pos=top;
      printf("Stack elements(top to bottom): ");
      while(pos!=NULL)
      {
         printf("%d",pos->data);
         pos=pos->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***

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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

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

#define MAX_SIZE 100

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

// Function to push a book ID onto the stack
void push(int bookID) {
    if (top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack[++top] = bookID;
    printf("Book ID %d is pushed onto the stack\n", bookID);
}

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

// Function to display book IDs in the stack
void display() {
    if (top == -1) {
        printf("Stack is empty\n");
        return;
    }
    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) {
        scanf("%d", &choice);

        switch (choice) {
            case 1: // Push book ID
                scanf("%d", &bookID);
                push(bookID);
                break;
            case 2: // Pop book ID
                pop();
                break;
            case 3: // Display stack contents
                display();
                break;
            case 4: // Exit program
                printf("Exiting the program\n");
                return 0;
            default: // Handle invalid choices
                printf("Invalid choice\n");
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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;
}

// You are using GCC
void push(char value) {
    //Type your code here
    if(top==MAX_SIZE-1){
        return;
    }
    items[++top]=value;
    printf("Pushed: %c\n",value);
}

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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;
}

// You are using GCC
int isOperand(char ch) {
    //type your code here
    return (ch>='a' && ch<='z') || (ch>='A' && ch<='Z') || (ch>='0' && ch<='9');
}

int Prec(char ch) {
   //type your code here
   switch(ch){
       case '+':
       case '-': return 1;
       case '*':
       case '/': return 2;
       case '^': return 3;
       default: return -1;
   }
```

```
}
int isRight(char ch){
    return ch=='^';

}
void infixToPostfix(char* exp) {
    //type your code here
    int i,k;
    int len=strlen(exp);
    char* result=(char*)malloc((len+1)*sizeof(char));

    struct Stack* stack=createStack(len);
    if(!stack) return;

    for(i=0,k=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);
            if(!isEmpty(stack) && peek(stack)!='('){
                free(stack->array);
                free(stack);
                free(result);
                return;
            }
            else{
                pop(stack);
            }
        }else{
            while(!isEmpty(stack) && ((Prec(exp[i]) < Prec(peek(stack))) || (Prec(exp[i])
== Prec(peek(stack)) && !isRight(exp[i])))){
                result[k++]=pop(stack);
            }
            push(stack, exp[i]);
        }
    }
    while(!isEmpty(stack))
```

```
        result[k++]=pop(stack);
    result[k]='\0';
    printf("%s\n",result);
}

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

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

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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){
        printf("Adding Section: %c\n ", value);
        struct Node* newnode=(struct Node*)malloc(sizeof(struct Node));

        newnode->data=value;

        newnode->next=top;

        top=newnode;
}
void pop() {

        if(top==NULL){
            printf("Stack is empty. Cannot pop. \n");
            return;
        }else{
        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;
    }
    else{
        printf("Enrolled Sections: ");
        struct Node* temp=top;
        while(temp!=NULL){
            printf("%c",temp->data);
            temp=temp->next;
        }
        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: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
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

You are required to implement a stack data structure using a singly linked list that follows the Last In, First Out (LIFO) principle.

The stack should support the following operations: push, pop, display, and peek.

Input Format

The input consists of four space-separated integers N, representing the elements to be pushed onto the stack.

Output Format

The first line of output displays all four elements in a single line separated by a space.

The second line of output is left blank to indicate the pop operation without displaying anything.

The third line of output displays the space separated stack elements in the same line after the pop operation.

The fourth line of output displays the top element of the stack using the peek operation.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 11 22 33 44
Output: 44 33 22 11

33 22 11
33

Answer

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

// Node structure
typedef struct Node {
    int data;
    struct Node* next;
} Node;

Node* top = NULL;

// Push operation
void push(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) return;

    newNode->data = value;
    newNode->next = top;
    top = newNode;
```

```
}

// Pop operation
void pop() {
    if (top == NULL) return;

    Node* temp = top;
    top = top->next;
    free(temp);
}

// Display operation
void display() {
    Node* current = top;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

// Peek operation
void peek() {
    if (top != NULL) {
        printf("%d\n", top->data);
    }
}

int main() {
    int a, b, c, d;
    scanf("%d %d %d %d", &a, &b, &c, &d);

    // Push in reverse to preserve input order (last goes on top)
    push(a);
    push(b);
    push(c);
    push(d);

    // First line output: full stack
    display();

    // Second line: blank (pop without output)
```

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

    // Third line: stack after pop
    display();

    // Fourth line: top element
    peek();

    return 0;
}
```

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

2. 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

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

#define MAX 100

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

// Push to stack
void push(char c) {
    if (top == MAX - 1) return;
    stack[++top] = c;
}

// Pop from stack
char pop() {
    if (top == -1) return '\0';
    return stack[top--];
}

// Peek at stack top
char peek() {
    if (top == -1) return '\0';
    return stack[top];
}

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

// Check if character is operator
```

```
int isOperator(char c) {
    return c == '+' || c == '-' || c == '*' || c == '/';
}

// Infix to Postfix Conversion
void infixToPostfix(char* infix) {
    char postfix[MAX];
    int j = 0;

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

        if (isdigit(ch)) {
            postfix[j++] = ch;
        }
        else if (ch == '(') {
            push(ch);
        }
        else if (ch == ')') {
            while (peek() != '(' && top != -1) {
                postfix[j++] = pop();
            }
            pop(); // remove '('
        }
        else if (isOperator(ch)) {
            while (top != -1 && precedence(ch) <= precedence(peek())) {
                postfix[j++] = pop();
            }
            push(ch);
        }
    }

    // Pop remaining operators
    while (top != -1) {
        postfix[j++] = pop();
    }

    postfix[j] = '\0';
    printf("%s\n", postfix);
}

int main() {
```

```
    char infix[31];
    scanf("%30s", infix); // limit input to 30 characters
    infixToPostfix(infix);
    return 0;
}
```

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

3. Problem Statement

Raj is a software developer, and his team is building an application that processes user inputs in the form of strings containing brackets. One of the essential features of the application is to validate whether the input string meets specific criteria.

During testing, Raj inputs the string "(([])){}". The application correctly returns "Valid string" because the input satisfies the criteria: every opening bracket (, [, and { has a corresponding closing bracket),], and }, arranged in the correct order.

Next, Raj tests the application with the string "([)]". This time, the application correctly returns "Invalid string" because the opening bracket [is incorrectly closed by the bracket), which violates the validation rules.

Finally, Raj enters the string "{[()]}". The application correctly identifies it as a "Valid string" since all opening brackets are matched with the corresponding closing brackets in the correct order.

As a software developer, Raj's responsibility is to ensure that the application works reliably and produces accurate results for all input strings, following the validation rules. He accomplishes this by using a method for solving such problems.

Input Format

The input comprises a string representing a sequence of brackets that need to be validated.

Output Format

The output prints "Valid string" if the string is valid. Otherwise, it prints "Invalid string".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: (([]))\{\}
Output: Valid string

Answer

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

#define MAX 100

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

void push(char c) {
    if (top == MAX - 1) return;
    stack[++top] = c;
}

char pop() {
    if (top == -1) return '\0';
    return stack[top--];
}

char peek() {
    if (top == -1) return '\0';
    return stack[top];
}

int isMatchingPair(char open, char close) {
    return (open == '(' && close == ')') ||
           (open == '{' && close == '}') ||
           (open == '[' && close == ']');
}
```

```
int isValid(char* str) {
    for (int i = 0; str[i]; i++) {
        char ch = str[i];
        if (ch == '(' || ch == '{' || ch == '[') {
            push(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            if (top == -1 || !isMatchingPair(pop(), ch)) {
                return 0;
            }
        }
    }
    return top == -1;
}

int main() {
    char input[101];
    scanf("%100s", input);

    if (isValid(input)) {
        printf("Valid string\n");
    } else {
        printf("Invalid string\n");
    }

    return 0;
}
```

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