Name:-Aakash Bhatt
Sap-id:- 500124633

Data Structure

Lab-8

Submitted by:

Aakash Bhatt

500124633

(2023-2024)

Submitted to:

Pankaj Sir

Name:-Aakash Bhatt
Sap-id:- 500124633

Write code to implement Stack Data structures. Implement the Push and Pop operation in the stack.

```c
#include <stdio.h>

#include <stdlib.h>

#define SIZE 4

int top = -1, stack[SIZE];
void push();
void pop();
void display();
void peek();

int main()
{
    int choice;

    while (1)
    {
        printf("\nPerform operations on the stack:");
        printf("\n1.Push the element\n2.Pop the
element\n3.Display\n4.Peek\n5.Exit");
        printf("\n\nEnter the choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            display();
            break;
        case 4:
            peek();
            break;
        case 5:
            exit(0);

        default:
            printf("\nInvalid choice!!");
        }
    }
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
}

void push()
{
    int x;

    if (top == SIZE - 1)
    {
        printf("\nOverflow!!");
    }
    else
    {
        printf("\nEnter the element to be added onto the stack: ");
        scanf("%d", &x);
        top = top + 1;
        stack[top] = x;
    }
}

void pop()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nPopped element: %d", stack[top]);
        top = top - 1;
    }
}

void display()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nElements present in the stack: \n");
        for (int i = top; i >= 0; --i)
            printf("%d\n", stack[i]);
    }
}
void peek()
{
    if (top == -1)
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
    {
        printf("Stack is empty!");
    }
    else
    {
        printf("Topmost element of the stack is: %d", stack[top]);
    }
}
```

```
PS D:\MCA\MCA-DSA\LAB-8> gcc .\Question1.c
PS D:\MCA\MCA-DSA\LAB-8> .\a.exe

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Display
4.Peek
5.Exit

Enter the choice: 1

Enter the element to be added onto the stack: 10

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Display
4.Peek
5.Exit

Enter the choice: 1

Enter the element to be added onto the stack: 20
```

Kindly use the stack data structure to reverse a number/string and if the number is a palindrome, print that number/string.

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

#define MAX_SIZE 100

// Define the stack data structure
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```c
struct Stack
{
    char items[MAX_SIZE];
    int top;
};

// Initialize the stack
void initialize(struct Stack *stack)
{
    stack->top = -1;
}

// Check if the stack is empty
int isEmpty(struct Stack *stack)
{
    return stack->top == -1;
}

// Check if the stack is full
int isFull(struct Stack *stack)
{
    return stack->top == MAX_SIZE - 1;
}

// Push a character onto the stack
void push(struct Stack *stack, char value)
{
    if (isFull(stack))
    {
        printf("Stack is full. Cannot push %c.\n", value);
        return;
    }
    stack->items[++stack->top] = value;
}

// Pop a character from the stack
char pop(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty. Cannot pop.\n");
        exit(1);
    }
    return stack->items[stack->top--];
}

int main()
{
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```c
    struct Stack stack;
    initialize(&stack);

    char input[MAX_SIZE];
    printf("Enter a string: ");
    scanf("%s", input);

    int length = strlen(input);
    int i;

    // Push each character onto the stack
    for (i = 0; i < length; i++)
    {
        push(&stack, input[i]);
    }

    char reversed[MAX_SIZE];
    int j = 0;

    // Pop characters from the stack to reverse the input
    while (!isEmpty(&stack))
    {
        reversed[j++] = pop(&stack);
    }

    reversed[j] = '\0'; // Null-terminate the reversed string

    printf("Reversed: %s\n", reversed);

    // Check if the input is a palindrome
    if (strcmp(input, reversed) == 0)
    {
        printf("%s is a palindrome!\n", input);
    }
    else
    {
        printf("%s is not a palindrome.\n", input);
    }

    return 0;
}
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
PS D:\MCA\MCA-DSA\LAB-8> gcc .\Question2.c
PS D:\MCA\MCA-DSA\LAB-8> .\a.exe
Enter a string: anshu
Reversed: uhsna
anshu is not a palindrome.
PS D:\MCA\MCA-DSA\LAB-8> .\a.exe
Enter a string: hllh
Reversed: hllh
hllh is a palindrome!
PS D:\MCA\MCA-DSA\LAB-8>
```

For the given string 2+3*5+8/2+6, convert this into postfix and eventually solve this using Stack.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h> // Include ctype.h for isdigit function

#define MAX_SIZE 100

// Define a stack data structure for characters
struct Stack
{
    char items[MAX_SIZE];
    int top;
};

// Initialize the stack
void initialize(struct Stack *stack)
{
    stack->top = -1;
}

// Check if the stack is empty
int isEmpty(struct Stack *stack)
{
    return stack->top == -1;
}

// Check if the stack is full
int isFull(struct Stack *stack)
{
    return stack->top == MAX_SIZE - 1;
}
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```c
// Push a character onto the stack
void push(struct Stack *stack, char value)
{
    if (isFull(stack))
    {
        printf("Stack is full. Cannot push %c.\n", value);
        return;
    }
    stack->items[++stack->top] = value;
}

// Pop a character from the stack
char pop(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty. Cannot pop.\n");
        exit(1);
    }
    return stack->items[stack->top--];
}

// Get the precedence of an operator
int getPrecedence(char operator)
{
    if (operator== '+' || operator== '-')
        return 1;
    if (operator== '*' || operator== '/')
        return 2;
    return 0; // Lower precedence for other characters (operands,
parentheses)
}

// Convert infix expression to postfix
void infixToPostfix(char *infix, char *postfix)
{
    struct Stack operatorStack;
    initialize(&operatorStack);

    int infixLength = strlen(infix);
    int postfixIndex = 0;

    for (int i = 0; i < infixLength; i++)
    {
        char currentChar = infix[i];

        if (isdigit(currentChar))
        {
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
            postfix[postfixIndex++] = currentChar; // Operand, add to
postfix
        }
        else if (currentChar == '(')
        {
            push(&operatorStack, currentChar);
        }
        else if (currentChar == ')')
        {
            while (!isEmpty(&operatorStack) &&
operatorStack.items[operatorStack.top] != '(')
            {
                postfix[postfixIndex++] = pop(&operatorStack);
            }
            pop(&operatorStack); // Pop and discard '('
        }
        else
        {
            // Operator
            while (!isEmpty(&operatorStack) && getPrecedence(currentChar)
<= getPrecedence(operatorStack.items[operatorStack.top]))
            {
                postfix[postfixIndex++] = pop(&operatorStack);
            }
            push(&operatorStack, currentChar);
        }
    }

    // Pop any remaining operators from the stack
    while (!isEmpty(&operatorStack))
    {
        postfix[postfixIndex++] = pop(&operatorStack);
    }

    postfix[postfixIndex] = '\0'; // Null-terminate the postfix string
}

// Evaluate a postfix expression
int evaluatePostfix(char *postfix)
{
    struct Stack operandStack;
    initialize(&operandStack);

    int postfixLength = strlen(postfix);

    for (int i = 0; i < postfixLength; i++)
    {
        char currentChar = postfix[i];
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
        if (isdigit(currentChar))
        {
            push(&operandStack, currentChar - '0'); // Convert char to
int and push as operand
        }
        else
        {
            int operand2 = pop(&operandStack);
            int operand1 = pop(&operandStack);

            switch (currentChar)
            {
            case '+':
                push(&operandStack, operand1 + operand2);
                break;
            case '-':
                push(&operandStack, operand1 - operand2);
                break;
            case '*':
                push(&operandStack, operand1 * operand2);
                break;
            case '/':
                push(&operandStack, operand1 / operand2);
                break;
            }
        }
    }

    return pop(&operandStack); // Result is on top of the operand stack
}

int main()
{
    char infixExpression[] = "2+3*5+8/2+6";
    char postfixExpression[MAX_SIZE];

    infixToPostfix(infixExpression, postfixExpression);
    printf("Infix Expression: %s\n", infixExpression);
    printf("Postfix Expression: %s\n", postfixExpression);

    int result = evaluatePostfix(postfixExpression);
    printf("Result: %d\n", result);

    return 0;
}
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
PS D:\MCA\MCA-DSA\LAB-8> gcc .\Question3.c
PS D:\MCA\MCA-DSA\LAB-8> .\a.exe
Infix Expression: 2+3*5+8/2+6
Postfix Expression: 235*+82/+6+
Result: 27
PS D:\MCA\MCA-DSA\LAB-8>
```

Implement a functionality where a dedicated memory which was allocated using Malloc is about to get filled, reallocate the memory if the available memory is less than 10 %.

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int *ptr;
    int i, n;
    printf("Enter the number of elements\n");
    scanf("%d", &n);
    printf("Entered no is %d", n);
    ptr = (int *)malloc(n * sizeof(int));
    if (ptr == NULL)
    {
        printf("Memeory not allocated\n");
        exit(0);
    }
    else if ()
        else
        {
            printf("Memory is allocated\n");
            for (i = 0; i < n; i++)
            {
                ptr[i] = i;
            }
            printf("Elemets in the array are:\n");
            for (i = 0; i < n; i++)
            {
                printf("%d", ptr[i]);
            }
        }
    return 0;
}
```