Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. 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***

2. 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***

3. 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***

4. 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

3

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

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

Answer

142

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

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

Answer

All of the mentioned options

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

7. Elements are Added on _____ of the Stack.

Answer

Top

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

8. 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***

9. 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***

10. 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***

11. 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***

12. 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***

13. 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***

14. 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***

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

Answer

Efficient memory usage

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

16. 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**

17. 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**

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

Answer

-18

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

19. 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***

20. 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***

Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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;


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);
}
```

```
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;
            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: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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

Buvi is working on a project that requires implementing an array-stack data structure with an additional feature to find the minimum element.

Buvi needs to implement a program that simulates a stack with the following functionalities:

Push: Adds an element onto the stack.Pop: Removes the top element from the stack.Find Minimum: Finds the minimum element in the stack.

Buvi's implementation should efficiently handle these operations with a maximum stack size of 20.

Input Format

The first line of input consists of an integer N, representing the number of

elements to push onto the stack.

The second line consists of N space-separated integer values, representing the elements to be pushed onto the stack.

Output Format

The first line of output displays "Minimum element in the stack: " followed by the minimum element in the stack after pushing all elements.

The second line displays "Popped element: " followed by the popped element.

The third line displays "Minimum element in the stack after popping: " followed by the minimum element in the stack after popping one element.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 2 8 1
Output: Minimum element in the stack: 1
Popped element: 1
Minimum element in the stack after popping: 2

Answer

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

#define MAX 20

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

// Push function
void push(int x) {
    if (top >= MAX - 1) {
        return;
    }
```

```
    stack[++top] = x;

    // Update min stack
    if (minTop == -1 || x <= minStack[minTop]) {
        minStack[++minTop] = x;
    }
}

// Pop function
int pop() {
    if (top == -1) {
        return -1;
    }
    int popped = stack[top--];

    if (popped == minStack[minTop]) {
        minTop--;
    }
    return popped;
}

int getMin() {
    if (minTop == -1) {
        return -1;
    }
    return minStack[minTop];
}

int main() {
    int N, i, val;

    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &val);
        push(val);
    }

    printf("Minimum element in the stack: %d\n", getMin());

    int popped = pop();
    printf("Popped element: %d\n", popped);
```

```
    printf("Minimum element in the stack after popping: %d\n", getMin());

    return 0;
}
```

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

2. 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 for the stack
struct Node {
    int data;
    struct Node* next;
};

// Top pointer
struct Node* top = NULL;

// Push operation
void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
}

// Pop operation
void pop() {
    if (top == NULL) return;
    struct Node* temp = top;
    top = top->next;
    free(temp);
}

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

```
    }
}

// Peek operation
int peek() {
    if (top == NULL) return -1;
    return top->data;
}

int main() {
    int a, b, c, d;

    // Reading four integers
    scanf("%d %d %d %d", &a, &b, &c, &d);

    // Pushing elements in reverse order to match LIFO stack behavior
    push(a);
    push(b);
    push(c);
    push(d);

    // Display stack after all pushes
    display();
    printf("\n");

    // Pop top element
    pop();

    // Display stack after pop
    display();
    printf("\n");

    // Peek at the top element
    printf("%d\n", peek());

    return 0;
}
```

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

3. Problem Statement

In an educational setting, Professor Smith tasks Computer Science students with designing an algorithm to evaluate postfix expressions efficiently, fostering problem-solving skills and understanding of stack-based computations.

The program prompts users to input a postfix expression, evaluates it, and displays the result, aiding students in honing their coding abilities.

Input Format

The input consists of the postfix mathematical expression.

The expression will contain real numbers and mathematical operators (+, -, *, /), without any space.

Output Format

The output prints the result of evaluating the given postfix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 82/
Output: 4

Answer

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

#define MAX 100

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

```
// Push a value onto the stack
void push(float val) {
    if (top >= MAX - 1) return; // Overflow protection
    stack[++top] = val;
}

// Pop a value from the stack
float pop() {
    if (top == -1) return 0; // Underflow protection
    return stack[top--];
}

// Evaluate postfix expression
float evaluatePostfix(char* expr) {
    int i = 0;
    char ch;
    float a, b, result;

    while ((ch = expr[i++]) != '\0') {
        if (isdigit(ch)) {
            push((float)(ch - '0'));
        } else {
            b = pop();
            a = pop();
            switch (ch) {
                case '+': result = a + b; break;
                case '-': result = a - b; break;
                case '*': result = a * b; break;
                case '/': result = a / b; break;
                default:
                    printf("Invalid operator: %c\n", ch);
                    return 0;
            }
            push(result);
        }
    }

    return pop();
}

int main() {
    char expr[MAX];
```

```
    // Read postfix expression (no spaces)
    scanf("%s", expr);

    // Evaluate and print result
    float result = evaluatePostfix(expr);

    // If the result is a whole number, print as integer
    if (fabs(result - (int)result) < 0.000001) {
        printf("%d\n", (int)result);
    } else {
        printf("%.2f\n", result); // Optionally, for real numbers
    }

    return 0;
}
```

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