Institute of Computer Technology
B. Tech. Computer Science and Engineering
Sub: DS Branch: BDA Class: A

Name: Jatin Patel
Enrolment No.: 23162121029
Sem: 3
Class: A
Subject: DS
Practical: 05
Date: 21/08/24

Practical 5:

Implement an Undo Mechanism in a Text Editor text editors, such as Microsoft Word and Google Docs, provide an &quot;undo&quot; feature that allows users to reverse their last actions, including typing and deleting text.
Design and implement an undo mechanism using a stack data structure to handle text editing actions.
Operations:
1. Push Operation: Whenever the user performs an action (e.g., typing or deleting text), store the action on an undo stack. Each action should be recorded with enough information to allow it to be reversed.
2. Pop Operation: When the user clicks &quot;undo,&quot; remove the most recent action from the stack. Reverse this action to restore the text to its previous state.
3. Peek Operation: Allow the system to inspect the most recent action on the stack without removing it. This operation can be used to display information about the next action that would be undone.

Demonstrate the undo functionality by simulating a sequence of user actions and corresponding undo operations, including the ability to peek at the next undo action.

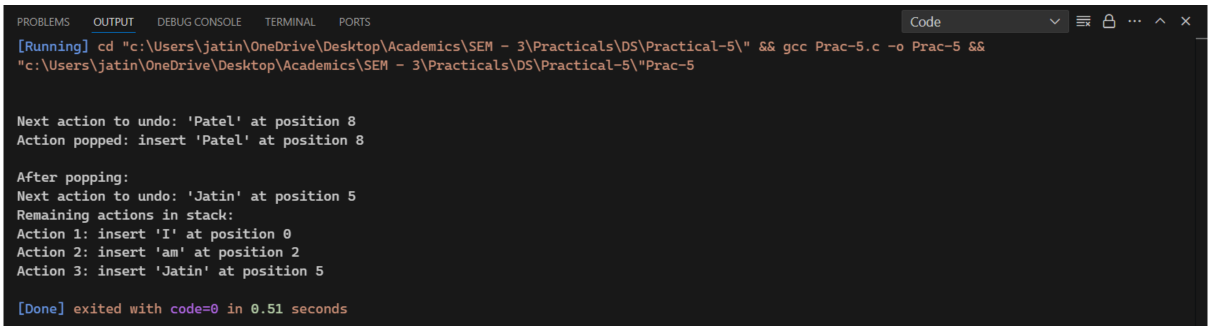
Expected Outcome:
A functional demonstration where a sequence of text editing actions can be undone in reverse order, showcasing the stack-based undo mechanism, with the ability to preview the next action to be undone using the peek operation.

Code:

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

#define MAX_ACTIONS 10
#define MAX_STRING_LENGTH 50

typedef struct {
    char action[10];
    char text[MAX_STRING_LENGTH];
    int position;
} UndoAction;

typedef struct {
    UndoAction actions[MAX_ACTIONS];
    int top;
} UndoStack;

void pushAction(UndoStack *stack, const char *action, const char *text,
int position) {
    if (stack->top >= MAX_ACTIONS - 1) {
        printf("Stack overflow. Cannot push any more actions.\n");
        return;
    }
    stack->top++;
    strcpy(stack->actions[stack->top].action, action);
    strcpy(stack->actions[stack->top].text, text);
    stack->actions[stack->top].position = position;
}

void popAction(UndoStack *stack) {
    if (stack->top == -1) {
        printf("Stack underflow. No actions to pop.\n");
        return;
    }
    UndoAction action = stack->actions[stack->top];
    stack->top--;
    printf("Action popped: %s '%s' at position %d\n", action.action,
action.text, action.position);
}

void peekAction(const UndoStack *stack) {
    if (stack->top == -1) {
        printf("No actions to undo.\n");
        return;
    }
    UndoAction action = stack->actions[stack->top];
```

```c
    printf("Next action to undo: '%s' at position %d\n", action.text,
action.position);
}

int main() {
    UndoStack stack;
    stack.top = -1;

    printf("\n\n");
    pushAction(&stack, "insert", "I", 0);
    pushAction(&stack, "insert", "am", 2);
    pushAction(&stack, "insert", "Jatin", 5);
    pushAction(&stack, "insert", "Patel", 8);

    peekAction(&stack);

    popAction(&stack);

    printf("\nAfter popping:\n");
    peekAction(&stack);

    printf("Remaining actions in stack:\n");
    for (int i = 0; i <= stack.top; i++) {
        printf("Action %d: %s '%s' at position %d\n", i + 1,
stack.actions[i].action, stack.actions[i].text,
stack.actions[i].position);
    }

    return 0;
}
```

Output:



```
[Running] cd "c:\Users\jatin\OneDrive\Desktop\Academics\SEM - 3\Practicals\DS\Practical-5\" && gcc Prac-5.c -o Prac-5 &&
"c:\Users\jatin\OneDrive\Desktop\Academics\SEM - 3\Practicals\DS\Practical-5\"Prac-5


Next action to undo: 'Patel' at position 8
Action popped: insert 'Patel' at position 8

After popping:
Next action to undo: 'Jatin' at position 5
Remaining actions in stack:
Action 1: insert 'I' at position 0
Action 2: insert 'am' at position 2
Action 3: insert 'Jatin' at position 5

[Done] exited with code=0 in 0.51 seconds
```