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:

o Whenever the user performs an action (e.g., typing or deleting text), store the

action on an undo stack.

o Each action should be recorded with enough information to allow it to be

reversed.

2. Pop Operation:

o When the user clicks &quot;undo,&quot; remove the most recent action from the stack.

o Reverse this action to restore the text to its previous state.

3. Peek Operation:

o Allow the system to inspect the most recent action on the stack without removing

it.

o 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* <stdlib.h>

#*include* <stdbool.h>

#*include* <limits.h>

#*define* *MAX\_SIZE* 100

typedef struct {

    int \*array;

    int top;

    int capacity;

} Stack;

Stack\* *createStack*(int capacity) {

    Stack\* stack = (Stack\*)*malloc*(sizeof(Stack));

    stack->capacity = capacity;

    stack->top = -1;

    stack->array = (int\*)*malloc*(stack->capacity \* sizeof(int));

*return* stack;

}

*bool* *isFull*(Stack\* stack) {

*return* stack->top == stack->capacity - 1;

}

*bool* *isEmpty*(Stack\* stack) {

*return* stack->top == -1;

}

void *push*(Stack\* stack, int item) {

*if* (*isFull*(stack)) {

*printf*("Stack Overflow! Cannot push %d\n", item);

*return*;

    }

    stack->array[++stack->top] = item;

*printf*("%d pushed to stack\n", item);

}

int *pop*(Stack\* stack) {

*if* (*isEmpty*(stack)) {

*printf*("Stack Underflow! Cannot pop from an empty stack\n");

*return* *INT\_MIN*;

    }

*return* stack->array[stack->top--];

}

int *redo*(Stack\* stack) {

*if* (*isEmpty*(stack)) {

*printf*("Stack Underflow! Cannot pop from an empty stack\n");

*return* *INT\_MIN*;

    }

*return* stack->array[stack->top++];

}

int *peek*(Stack\* stack) {

*if* (*isEmpty*(stack)) {

*printf*("Stack is empty\n");

*return* *INT\_MIN*;

    }

*return* stack->array[stack->top];

}

void *display*(Stack\* stack) {

*if* (*isEmpty*(stack)) {

*printf*("Stack is empty\n");

*return*;

    }

*printf*("Stack elements: ");

*for* (int i = stack->top; i >= 0; i--) {

*printf*("%d ", stack->array[i]);

    }

*printf*("\n");

}

int *size*(Stack\* stack) {

*return* stack->top + 1;

}

void *clear*(Stack\* stack) {

    stack->top = -1;

*printf*("Stack cleared\n");

}

void *freeStack*(Stack\* stack) {

*free*(stack->array);

*free*(stack);

}

int *main*() {

    Stack\* stack = *createStack*(*MAX\_SIZE*);

    int choice, item;

*while* (1) {

*printf*("\nStack Operations:\n");

*printf*("1. Push\n2. Pop\n3. Peek\n4. Display\n5. Size\n");

*printf*("6. Is Empty\n7. Is Full\n8. Redo\n9. Clear\n10. Exit\n");

*printf*("Enter your choice: ");

*scanf*("%d", &choice);

*switch* (choice) {

*case* 1:

*printf*("Enter the item to push: ");

*scanf*("%d", &item);

*push*(stack, item);

*break*;

*case* 2:

                item = *pop*(stack);

*if* (item != *INT\_MIN*)

*printf*("Popped item: %d\n", item);

*break*;

*case* 3:

                item = *peek*(stack);

*if* (item != *INT\_MIN*)

*printf*("Top item: %d\n", item);

*break*;

*case* 4:

*display*(stack);

*break*;

*case* 5:

*printf*("Stack size: %d\n", *size*(stack));

*break*;

*case* 6:

*printf*("Is stack empty? %s\n", *isEmpty*(stack) ? "Yes" : "No");

*break*;

*case* 7:

*printf*("Is stack full? %s\n", *isFull*(stack) ? "Yes" : "No");

*break*;

*case* 8:

                item = *redo*(stack);

*printf*("Redo item: %d\n", item);

*break*;

*case* 9:

*clear*(stack);

*break*;

*case* 10:

*freeStack*(stack);

*printf*("Exiting the program. Goodbye!\n");

*exit*(0);

*default*:

*printf*("Invalid choice. Please try again.\n");

        }

    }

*return* 0;

}

Output:

