```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define CAPACITY 1000

struct stack
{
    int data;
    struct stack *next;
} *top;

int size = 0;

void push(int element);
int pop();
void display();


int main()
{
    int choice, data;
    while(1)
    {
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        scanf("%d", &choice);

        switch(choice)
        {
            case 1:
                printf("Enter data to push into stack: ");
                scanf("%d", &data);
                push(data);
                break;

            case 2:
                data = pop();

                // If stack is not empty
                if (data != INT_MIN)
                    printf("Data => %d\n", data);
```

```
                break;

            case 3:
                display();
                break;

            case 4:
                exit(0);
                break;

            default:
                printf("Invalid choice\n");
        }

        printf("\n\n");
    }
    return 0;
}

void push(int element)
{
    if (size >= CAPACITY)
    {
        printf("Stack Overflow\n");
        return;
    }
    struct stack * newNode = (struct stack *) malloc(sizeof(struct
stack));
    newNode->data = element;
    newNode->next = top;
    top = newNode;
    size++;
    printf("Data pushed to stack.\n");
}

int pop()
{
    int data = 0;
    struct stack * topNode;
    if (size <= 0 || !top)
    {
        printf("Stack is empty.\n");
        return INT_MIN;
```

```
    }
    topNode = top;
    data = top->data;
    top = top->next;
    free(topNode);
    size--;
    return data;
}

void display()
{
    if(size<=0||!top)
    {
        printf("Stack is empty\n");
        return;
    }
    struct stack *current=top;
    printf("Stack elements: ");
    while(current!=NULL)
    {
        printf("%d  ",current->data);
        current=current->next;
    }
    printf("\n\n");
}
```

```
PS C:\Users\kadab\OneDrive\Desktop\DS> .\a.exe
1. Push
2. Pop
3. Display
4. Exit
1
Enter data to push into stack: 1
Data pushed to stack.


1. Push
2. Pop
3. Display
4. Exit
1
Enter data to push into stack: 2
Data pushed to stack.


1. Push
2. Pop
3. Display
4. Exit
1
Enter data to push into stack: 3
Data pushed to stack.


1. Push
2. Pop
3. Display
4. Exit
3
Stack elements: 3  2  1



1. Push
2. Pop
3. Display
4. Exit
2
Data => 3


1. Push
2. Pop
3. Display
4. Exit
3
Stack elements: 2  1



1. Push
2. Pop
3. Display
4. Exit
4
PS C:\Users\kadab\OneDrive\Desktop\DS>
```

void push (int elem
int pop();
display()
int main()
{
int choice, data;
while(1){

N 29 May
Emtd
SLL
SLL
Stacks, Queues, reverse, sort
concatenation