```cpp
//Bernard J. Gole Cruz, CS 202-2002, Assignment 6, problem 2
//This program implement stack class tempate with exception handling
#include <iostream>
#include <stdlib.h>

using namespace std;

//stack class template
template <class T>
class Stack{
    public:
        //stack operation
        void push(T value);
        T pop();
        void disp();
        bool isempty();
        bool isfull();

        //default constructor
        Stack();
        //parameterize constructor()
        Stack(int);
        //destructor
        ~Stack();


    private:
        int stackPointer;      //top of the stack
        T *element = NULL;     //pointer to stack array
        int stacksize;         //stack size




};
//function prototype and definitions

//default constructor set size to 0
template <class T>
Stack<T>::Stack(){

this->stacksize = -1;
stackPointer = 0;
}

//parameterized constructor
//deallocate size based on given parameter
template <class T>
Stack<T>::Stack(int stacksize){

element = new T [stacksize];
this->stacksize = stacksize;
stackPointer = 0;
}


//destructor, deallocate elements
template <class T>
Stack<T>::~Stack(){

delete []element;
}

//add element in stack
template <class T>
 void Stack<T>::push(T value){
     //check if stack is full
     if (isfull()){
         cout <<"can't push in a full stack" <<endl;
         cout <<"error operating the stack at position " << stackPointer << endl;
     }
     //add element into stack
     else{
        cout <<"Enter value: " ;
        cin >> value;

        //clear screen at every iteration
        system("CLS");
        //add item into stack and increment index
        element[stackPointer] = value;
        stackPointer++;
     }
 }

//remove element in stack
template <class T>
T Stack<T>::pop(){
    //check if stack is empty
    if (isempty()){
        cout <<"can't pop from empty stack" << endl;
        cout <<"error operating the stack at position " << stackPointer << endl;
        return -1;
    }
    //remove item from stack
    else{
        stackPointer--;
        return element[stackPointer];
    }
}

//check if stack is empty
template <class T>
bool Stack<T>::isempty(){

    return (stackPointer == 0);
 }

//check if stack is full
template <class T>
bool Stack<T>::isfull(){

    return (stackPointer == stacksize);
}

template <class T>
void Stack<T>::disp(){

    if(isempty()){
    cout <<"Stack: " ;
    }
    else{
    cout <<"Stack: " ;
    for (int i=0; i<stackPointer; i++){
        cout << element[i];
         if (i != (stackPointer - 1) ){
            cout << ",";
            }
        }
    }
}

//menu
```

```cpp
void menu(){
    //display choices in menu
    cout << "Menu: " << endl;
    cout << "1. push Element" << endl;
    cout << "2. pop element" << endl;
    cout << "3. exit" << endl;

};

//prompt user
template <class T>
void prompt(Stack<T> &obj,int &choice)
{
    //keep prompting if user choose number outside the menu
    try{
    cout <<endl;
    menu();
       cout << "Enter: ";
       if(!(cin >>choice) ){
            cin.clear();
            cin.ignore(100,'\n');
            throw choice;
                    }
        }
       catch(int choice){
            throw;
            }

    int value;
    //choices in menu
    switch (choice){
    case 1:
        //add elements in stack
        obj.push(value);
        cout << endl;
        //display stack contents
        obj.disp();
        break;

    case 2:
        //remove elements from stack
        obj.pop();
        cout << endl;
        //display stack contents
        obj.disp();
        break;

    case 3:
        //exit menu
            exit(0);

        default:
            //keep prompting until a correct choice is made
            cout << "try again!!" << endl;
        }
};


int main(){

//variables, declared object
int select;
Stack<int> number(4);
bool success = false;
cout <<endl;
cout <<"Stack:";

//will keep prompting if choice 3 is not press
//exception handling
while(true){
    try{
        prompt(number, select);
        success = true;
       }
    catch(...){
        }
    }
    return 0;

}
```