```
//:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
//                    Templated Stack Class - Joel Cressy
//:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#pragma once
template <class T>
class stacktype
{
public:
    stacktype();
    stacktype(T);
    void push(T);
    void pop();
    T gettop() { return top->data; }
    bool isEmpty() { return top == NULL; }
    int size() { return stacksize; }

private:
    struct linkstack {
        T data;
        linkstack *next;
    };
    linkstack *top;
    int stacksize;
};

template <class T>
stacktype<T>::stacktype()
{
    top = NULL;
    stacksize = 0;
}

template <class T>
stacktype<T>::stacktype(T data)
{
    top = new linkstack({data, NULL});
    stacksize = 1;
}

template <class T>
void stacktype<T>::push(T data)
{
    if(!isEmpty())
    {
        linkstack *pushed;
        pushed = new linkstack({top->data,top->next});
        top->data = data;
        top->next = pushed;
    }
    else
    {
        top = new linkstack;
        top->data = data;
        top->next = NULL;
    }
    stacksize++;
}

template <class T>
void stacktype<T>::pop()
{

    linkstack *newtop = new linkstack;
    if (top->next != NULL) {
        newtop = top->next;
        top->data = newtop->data;
        top->next = newtop->next;
    }
    else {
        top = NULL;
    }
    delete newtop;
    stacksize--;
}
```