Ejercicio de prácticas de EDA del 2014.03.11

Pilas y notación polaca inversa

OBJETIVO
Usar el TAD Pila visto en clase para implementar una calculadora RPN.

ENTREGA
Entra en http://bantu.fdi.ucm.es/domjudge/team con tu nombre de grupo y contraseña. Sube un único .cpp como solución al problema (el .cpp debe compilar por sí mismo, sin #includes a excepción de <iostream>) al sistema, y una vez que funcione, pásate por el puesto del profesor para asegurarte de que tu solución es válida. *Envía algo antes de la clase, aunque no funcione, para optar a un 3.*

1. Enunciado
Te han pedido escribir un emulador de la famosa HP-48, la mejor calculadora programable de su momento, con la que se entrenaron miles de grandes ingenieros. Una calculadora sin igual (literalmente: cuando usas RPN, no hace falta usarlo).

2. La entrada
La entrada se leerá por entrada estándar (cin), y consistirá de múltiples líneas que pueden ser un número entero o un único carácter. Los números enteros se deben apilar. Los caracteres pueden ser **+**, **-**, *, **/** (operadores binarios, que sacan dos números de la pila e insertan el resultado de hacer la operación correspondiente), o **$** (que significa “mostrar la cima de la pila”) ó **!** (que significa “salir”).

3. La salida
Se deberá genera una única línea por instrucción **$** encontrada. Esa línea deberá mostrar la cima de la pila, o la palabra **Error.** si la última operación era inválida (por ejemplo, dividir entre cero ó operar sin suficientes operandos). Las operaciones inválidas no deben modificar la pila.

4. Entrada de ejemplo
```
5
1
$
2
+
4
*
+
3
-
```

$
!

5. Salida de ejemplo

1
14

6. Indicaciones adicionales

Respeta los formatos de entrada y salida al pie de la letra. No es lo mismo escribir “1” que escribir “1 ” – el espacio del final haría que el comprobador automático marque la respuesta como inválida.

El profesor verificará manualmente todas las respuestas para ver si contienen o no una una variable de tipo Pila, usando el TAD visto en clase. Cualquier implementación que no lo incluya será invalidada.

Recomendamos utilizar strings para entrada/salida:

```
#include <cstdlib>
#include <string>
...
string s;
cin >> s;
if (s[0] == '+') {
        // ...
} else if ( s[0] == '-') {
        // ... otros simbolos
} else {
        int n = atoi(s.c_str());
        // un numero que puedo apilar
}
```