Trabalho Prático I
Para 26 de Março, até as 11:59pm

Neste trabalho você irá escrever um pequeno programa em Cool. O objetivo é que você adquira familiaridade com a linguagem. Este trabalho é *individual*.

Uma máquina cujo armazenamento é constituido apenas por uma pilha é chamada de máquina de pilha. Considere a seguinte linguagem, bastante primitiva, para programar uma máquina de pilha:

Comando	*Resultado*
int	Empilha (push) o inteiro *int*
+	Empilha '+'
s	Empilha 's'
e	Avalia o topo da pilha (veja abaixo)
d	Exibe o conteúdo da pilha
x	Para

O comando 'd' exibe o conteúdo da pilha, um elemento por linha, começando pelo topo da pilha. O comportamento do comando 'e' depende do conteúdo da pilha quando ele é executado:

- Se '+' está no topo da pilha, então o '+' e os próximos dois inteiros da pilha são desempilhados e a soma deles é empilhada.
- Se 's' está no topo da pilha, então o 's' é desempilhado e os dois itens seguintes da pilha são trocados entre si.
- Se um inteiro está no topo da pilha ou a pilha está vazia, a pilha permanece inalterada.

Os seguintes exemplos mostram o efeito do comando 'e' em várias situações; o topo da pilha está a esquerda:

pilha antes	*pilha depois*
$+\ 1\ 2\ 5\ s\ldots$	$3\ 5\ s\ldots$
$s\ 1\ +\ +\ 99\ldots$	$+\ 1\ +\ 99$
$1\ +\ 3\ldots$	$1\ +\ 3\ldots$

Você deve implementar um interpretador para essa linguagem em Cool. A entrada do programa é uma série de comandos, um comando por linha. Seu interpretador deve utilizar `>` como prompt de comando (veja a seção **Exemplo de Compilação e Execução**). Seu programa não precisa fazer nenhum tratamento de erro, você pode assumir que todos os comandos são válidos e que o tipo e a quantidade de argumentos na pilha está correta. Você pode assumir que os inteiros são sem sinal. Seu interpretador deve terminar naturalmente; não chame `abort()` após receber um `x`.

Você é livre para implementar este programa em qualquer estilo que desejar. Entretanto, recomendamos que você tente desenvolver uma solução orientada a objetos para que se prepare melhor para os trabalhos futuros. Uma abordagem seria definir uma classe `StackCommand` que representa um comando genérico e, a partir dela, definir uma subclasse para cada comando específico. Sinta-se a vontade para utilizar os códigos de exemplo que foram disponibilizados, em particular atoi.cl para realizar conversão de strings para inteiros.

Exemplo de Compilação e Execução

A seguir está um exemplo de compilação e execução da solução:

```
%coolc stack.cl atoi.cl -o stack.s
%spim -file stack.s
SPIM Version 6.5 of January 4, 2003
Copyright 1990-2003 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /home/grad/ccomp/11/pedro.caldeira/students/compilers/lib/trap.handler
>1
>+
>2
>s
>d
s
2
+
1
>e
>e
>d
3
>x
COOL program successfully executed
```

Instruções adicionais

Juntamente com o enunciado, está sendo distribuída uma pasta que contém:

1. **examples:** uma lista de exemplos de programas escritos em Cool para que se familiarizem com a linguagem.
2. **cool-manual.pdf:** especificação da linguagem Cool.
3. **coolc:** um link simbólico para o compilador de Cool. Deve ser executado a partir das máquinas da graduação do DCC.
4. **spim:** um link simbólico para o simulador do MIPS. Deve ser executado a partir das máquinas da graduação do DCC.
5. **stack.cl:** arquivo que irá conter a sua implementação.
6. **atoi.cl:** arquivo que contém funções já prontas para conversão entre strings e inteiros.

Submissão

Simplesmente envie o arquivo stack.cl com sua implementação. Iremos compilar ele juntamente com atoi.cl, então, caso o utilizem, não é necessário enviá-lo.

Você pode testar sua implementação utilizando os links simbólicos do compilador e simulador que foram disponibilizados - vide **Exemplo de Compilação e Execução**. Lembre-se que estes links simbólicos só funcionam quando executados a partir das máquinas da graduação do DCC.

Atenção: Siga a formatação especificada a rigor, pois seu programa será avaliado comparando-se a saída dele com a da implementação correta. Vide, novamente, a seção **Exemplo de Compilação e Execução**.