Considere a express�o

E(n) = 5 \* n + E(n-2) \* 1 - 2 \* E(n-1)

para a qual definimos explicitamente E(0) = 1, E(1) = 2 e E(2) = 1. Supondo que queremos calcular o valor de n = 6, o arquivo de entrada � o seguinte, usando \*\*\*\*\* para indicar in�cio e final do arquivo (veja a descri��o detalhada do formato de arquivo no enunciado do EP)

\*\*\*\*\*

2

1

2

1

5 n \* -2 1 \* + 2 -1 \* -

6

\*\*\*\*\*

Seu EP usa uma pilha para controlar as chamadas recursivas, e para cada chamada recursiva, uma pilha espec�fica para o c�lculo da express�o. A pilha das chamadas recursivas empilha "contextos", as pilhas de c�lculo empilham inteiros.

Contextos representam as vari�veis envolvidas em cada chamada recursiva (voc� dever� decidir no seu EP quais s�o). Vamos assumir as seguintes vari�veis: ponteiro para a pilha de c�lculo, que vou chamar genericamente de pc, posi��o atual na lista que representa a express�o, que vou chamar genericamente de pa (ou seja, pa � um ponteiro para um n� da lista), e valor da vari�vel n, que vou chamar simplesmente de n. Assim, um contexto pode ser representada por uma struct contendo tr�s campos: ponteiro para vetor de inteiros (pilha de c�lculo), ponteiro para n� de lista (posi��o atual do c�lculo na lista da express�o) e inteiro (valor do argumento). A pilha da recurs�o � assim uma pilha de structs.

Vamos observar o que acontece no in�cio da execu��o do seu programa para o exemplo acima. Sua pilha de recurs�o est� vazia. O valor de n � 6. O valor de pa � o endere�o do n� que armazena 5, e aloca-se uma pilha pc de inteiros para o c�lculo (vetor de tamanho m�ximo N definido como uma constante no seu programa). Neste momento empilha-se a struct para esses tr�s valores na pilha de recurs�o. Agora, seu programa percorre a express�o da esquerda para a direita para realizar o c�lculo da express�o. Primeiro, empilha (na pilha de c�lculo) 5, que � o valor do primeiro n�, em seguida empilha 6 (valor de n), e quando chega em \*, desempilha 5 e 6 e empilha o resultado de 5 \* 6. Durante essa leitura, pa � atualizado para o n� correspondente da lista. Agora, seu programa chega em -2, que representa uma chamada recursiva E(n-2). Neste momento, seu programa "congela" o c�lculo da express�o para n = 6 (ou seja, deixa o estado atual do c�lculo empilhado na pilha de recurs�o) e inicia um novo c�lculo, com uma nova pilha de c�lculo pa, para n = 4 (6 - 2), a partir do in�cio da express�o. Ent�o, seu programa cria uma nova pilha de c�lculo e empilha um novo contexto para esse novo c�lculo que se inicia.

Resumindo: sempre que seu programa encontra uma chamada recursiva na express�o, empilha um novo contexto, e inicia um novo c�lculo. Quando termina o c�lculo, desempilha o contexto, e continua o c�lculo no contexto imediatamente abaixo (que agora � o novo tipo da pilha de recurs�o), usando como valor da chamada recursiva o resultado rec�m-calculado.

Sempre que uma chamada recursiva � encontrada, ou seja, que um novo contexto � empilhado, seu programa imprime a pilha de recurs�o inteira (ou seja, mostra contexto por contexto, a partir do topo da pilha, um contexto por linha - o formato voc� decide, mas precisa ser leg�vel - mostrando para cada contexto os valores de pc, pa e n).

Quando seu programa termina o c�lculo da express�o, imprime o resultado. Se em algum instante a pilha de calculo estoura, seu programa imprime uma mensagem e para.

Para ajudar, eis uma refer�ncia para o c�lculo do valor de uma express�o na nota��o posfixa, usando uma pilha de c�lculo: http://www.ime.usp.br/~pf/mac0122-2002/aulas/stacks.html � a refer�ncia no enunciado do EP, agora com o link corrigido).