BCC202 – Estruturas de Dados I (2020-01)

Departamento de Computação - Universidade Federal de Ouro Preto - MG

AULA PRÁTICA 04 – RECURSIVIDADE

- Data de entrega: Até 26 de fevereiro às 17h..

- Procedimento para a entrega:.

1. Submissão: via **RunCodes**.
2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
4. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos *.h* e *.c* sempre que cabível.
5. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via **RunCodes**.
6. Dentre os arquivos submetidos, deve existir um intitulado *compilcao.txt*, contendo os comandos especificados no *prompt/console* para compilar e executar seu programa.
7. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.

- Bom trabalho!

Importante

Com a finalidade de facilitar o reuso e proporcionar a modularização do seu código, para os exercícios a seguir, especifique e implemente uma biblioteca de funções recursivas. Em outras palavras, as especificações das suas funções devem estar num arquivo *.h* e as suas implementações em um arquivo *.c*. Para testar sua solução, será necessário implementar o *main*() noutro arquivo (e.g., *main.c*). Observe que NÃO será necessário definir Tipos Abstratos de Dados.

Questão 01

Os números de *Fibonacci* são definidos pela seguinte recorrência[1]

- fib(0) = 0
- fib(1) = 1
- fib(n) = fib(n-1)+fib(n-2)

Mas não estamos interessados apenas nos números de *Fibonacci* aqui. Gostaríamos de saber também **quantas chamadas recursivas** seriam necessárias para calcular um determinado número de Fibonacci n, seguindo a recorrência normal. Uma vez que os números serão bem grandes, não será uma tarefa muito simples para você.

Exemplo de Entrada e Saída

A seguir, um exemplo de cenário de execução da sua função. A entrada é composta primeiramente por "n, deseja-se calcular o $n-esimo$ termo da séria de Fibonnaci. Ainda sobre a entrada, -1 indica que será inserido um próximo número para a função e -2" que não haverá novas entradas. A saída é composta, para cada n inserido, por dois números: $fib(n)$ e $iter$, este último é o número de chamadas recursivas realizadas.

[1] Exercício adaptado de: `https://www.urionlinejudge.com.br/judge/pt/problems/view/1033`.

Entrada	Saída
0 -1 1 -1 2 -1 4 -1 20 -1 40 -2	0 0 1 0 1 2 3 8 6765 21890 102334155 331160280

Questão 02

Implemente uma função recursiva para imprimir, na ordem inversa, os elementos de um vetor com *n* elementos.

Entrada e Saída

A entrada é composta do número de elementos do vetor seguidos dos seus elementos. A saída, será a impressão dos elementos do vetor em ordem inversa.

Exemplo de Entrada e Saída

A seguir, um exemplo de cenário de execução da sua função.

Entrada	Saída
6 2 4 6 8 10 12	12 10 8 6 4 2

Questão 03

Um truque bem conhecido para descobrir se um inteiro N é um múltiplo de nove é computar a soma S dos seus dígitos[2]. Se S é um múltiplo de nove, então N também é. Este é um teste recursivo e a profundidade da recursão necessária para obter a resposta para o número N é chamada o $grau-9$ de N. Assim, por exemplo, considere o número 99999999999999999999:

- 9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9+9=189
- 1+8+9=18
- 1+8=9
- **Grau-9** = 3

Sua tarefa é, dado um inteiro positivo N, de maneira recursiva, determinar se ele é um múltiplo de nove e, caso ele seja, qual o seu $grau-9$.

[2]Exercício baseado na especificação disponível em `https://br.spoj.com/problems/RUMO9S/`

Etntrada

A entrada é composta inicialmente por um número N cujo $grau - 9$ será calculado. A entrada -1 indica que será submetido outro número para cálculo do $grau - 9$ e -2 indica que não será fornecida outra entrada.

Saída

A saída do programa deve indicar para cada N da entrada o seu $grau - 9$.

Exemplo de Entrada e Saída

A seguir, um possível cenário de execução da sua função.

Entrada	Saída
999999999999 -1 9 -1 99999998 -2	2 1 0

Questão 04

"A pesquisa ou busca binária (em inglês, *binary search algorithm*) é um algoritmo de busca em vetores que segue o paradigma de divisão e conquista. Ela parte do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor"[3].

Seu desafio é implementar de forma recursiva a busca binária.

Entrada

A entrada será composta incialmente por n, o número de elementos do vetor; seguido dos n elementos do vetor (que deve estar ordenado), e, por fim, k, a chave a ser buscado. A entrada poderá ser ainda -1 para realizar outra busca em outro vetor ou -2, indicando o fim da entrada.

Saída

A saída será composta, para cada busca realizada, pelo índice i da posição em que o elemento foi encontrado ou -1, caso o elemento não exista.

Exemplo de Entrada e Saída

A seguir, um exemplo de cenário de execução da busca binária.

[3] Definição extraída de `https://pt.wikipedia.org/wiki/Pesquisa_binÃąria`

Entrada	Saída
4 6 2 4 8 16 32 64 64 -1 5 1 2 3 4 5 6 -1 5 2 4 6 8 10 10 -2	5 -1 4

Submissão Final

Como mencionado no cabeçalho deste arquivo, deve existir somente um *main*() para testar a implementação de cada uma das quatro funções recursivas. Assim sendo, o arquivo de entrada será único, contendo 1, 2, 3 e 4 para executar as soluções, respectivamente, das questões 1, 2, 3 e 4. A entrada -100 indica que o programa deve ser encerrado. De maneira similar, o arquivo de saída também será único. A seguir, um exemplo de cenário de execução de todo o programa. Observe que as entradas e saídas de cada questão são as mesmas listadas anteriormente.

Entrada	Saída
1 0 -1 1 -1 2 -1 4 -1 20 -1 40 -2 2 6 2 4 6 8 10 12 3 999999999999 -1 9 -1 99999998 -2 4 6 2 4 8 16 32 64 64 -1 5 1 2 3 4 5 6 -1 5 2 4 6 8 10 10 -2 -100	0 0 1 0 1 2 3 8 6765 21890 102334155 331160280 12 10 8 6 4 2 3 1 0 5 -1 4