Lista extra de exercícios para a P2

Recursão; TAD;
Vetores e matrizes;
Strings

Oficina de Programação em C (ICP037)

Prof. Ronald Souza
IC/UFRJ

Questão 1)

a) Um palíndromo é essencialmente uma palavra ou número que, ao se ler "de trás para frente", permanece inalterada. Por exemplo: "radar", "reviver", 1331, 90209, etc. Escreva um programa em C que leia **uma string** do teclado e determine se a mesma trata-se de um palíndromo.

b) Escreva um programa que inverta uma string qualquer de entrada (do teclado), e imprima a string invertida.
Por exemplo: para a string de entrada *"Professor"* (9 caracteres + caractere nulo), deverá ser impresso *"rosseforP"* pela simples leitura **sequencial**, a partir do **índice 0** da string **já invertida** até o índice 9 da mesma.

c) Escreva um programa que leia uma string do teclado e chame uma função "totalLetra" que receberá a string de entrada e também uma letra qualquer e contará quantas vezes essa letra ocorre no texto lido. A assinatura da função será:

```
int totalLetra(char[] texto, char letra);
```

Imprima o total na função **main**.

Questão 2) Matematicamente, pode-se definir o Máximo Divisor Comum (MDC) entre dois inteiros positivos **a** e **b** como

```
mdc(a, b) = a           , se b = 0;
          = mdc(b, a%b) , caso contrário
```

Podemos assim escrever uma **função recursiva** para o cálculo de MDC, como segue:

```
int mdc(int x, int y){
    if (y == 0) //caso base
        return x;
    return mdc(y, x % y); //chamada recursiva!
}
```

Ou simplesmente:

```
int mdc(int x, int y){
    return (y == 0) ? x : mdc(y, x % y);
}
```

Agora é a sua vez:

a) Vimos que a quantidade de dígitos de um inteiro ‘x’ pode ser obtida iterativamente pela seguinte função (onde ‘x’ é o parâmetro de entrada):

```
int numDigitos(int x) {
    int num = 0;
    do {
       num++;    //x possui ao menos 1 dígito.
       x /= 10;
    } while (x); //Equivale a "while(x != 0)"
    return num;
}
```

Escreva uma versão recursiva da função numDigitos acima.

b) Vimos que a soma dos *n* primeiros termos de uma série harmônica produz o chamado **número harmônico** H_n definido abaixo

$$H_n = \sum_{k=1}^{n} \frac{1}{k}.$$

Escreva um programa em C que calcula **recursivamente** o número *Hn*, para algum N de entrada tal que N <= 100. **Teste se os valores encontrados pelo seu programa estão corretos.** São exemplos de saídas esperadas:

N	**Saída**
1	1
2	1.5
3	~1.83333
4	~2.08333
5	~2.28333
10	~2.92897
15	~3.31823
20	~3.59774

Questão 3) Implemente um programa que cumpra os seguintes requisitos:

- Crie o tipo de dado **Polinomio**, fixado em grau 2 (isto é, todos os polinômios serão de 2o grau), e use um vetor de tamanho 3 para guardar os coeficientes (o quadrático, o linear e o independente);
- Leia do teclado os coeficientes de dois polinômios, p1 e p2, desse tipo.
- Imprima “Verdadeiro” se, para um inteiro $N \geq 2$, é o caso que p1 = *N* x p2 (ou vice-versa), e “Falso” caso contrário. Por exemplo:

Entrada:
p1 = {9, 12, 21}
p2 = {3, 4, 7}

Saída:
“Verdadeiro” //pois $9x^2 + 12x + 21 == \mathbf{3} \times (3x^2 + 4x + 7)$, para ***N* = 3.**

Questão 4) JOGO DE CARTAS

Considere um jogo de cartas para N jogadores onde

- são utilizadas somente as cartas que são numeradas, ou seja, as cartas de 2 até 10;
- uma carta vale, em pontos, exatamente o seu número se o seu naipe não for copas.
- cartas de copas valem duas vezes o seu número. Por exemplo, um "3 de espadas" vale 3 pontos; um "3 de copas" vale 6 pontos.
- a cada rodada, um jogador por vez, do primeiro ao N-ésimo, revelará uma única carta para os demais jogadores.
- o jogador cuja carta revelada é a de **maior valor** recolherá todas as N cartas reveladas **naquela rodada** e conquistará seus respectivos pontos.
- uma carta já revelada **não pode ter sua pontuação repetida** por outro jogador. Por exemplo, se o primeiro jogador revela um "8 de ouros", sua carta vale 8 pontos. Portanto, nenhum outro jogador subsequente poderá, naquela rodada, revelar outra carta de 8 pontos (nesse caso, "8 de espadas", "8 de paus" ou "4 de copas").

O jogo será transmitido online e deseja-se prover ao público diversas informações a cada rodada.

a) Crie o tipo estruturado ***carta***, que terá 2 campos: um char referente ao naipe e um int para o seu número. Seu naipe será um dentre 'o', 'e', 'c', 'p' (ouros, espadas, copas e paus, respectivamente).
b) crie o tipo estruturado ***jogador***, que terá 3 campos: um inteiro referente ao seu **ID**, uma string (char*) referente ao seu **nome** e um vetor de 5 cartas, referente à sua **mão**.
c) Crie uma função que receba uma ***carta*** de entrada e retorne 1 caso tal carta seja válida e 0 caso contrário. Uma carta é válida se o seu número está entre 2 e 10 ***e também*** o seu naipe é um dos listados no item “a”.
d) Crie uma função que receba duas cartas de entrada e retorne 1 caso estas sejam idênticas ou 0 caso contrário. *Duas cartas são consideradas idênticas **apenas** se tanto o naipe quanto o número são iguais*.
e) Crie uma função que receba 2 jogadores de entrada e retorne o total de cartas em comum entre eles. ***Use a função do item d como auxiliar nessa tarefa.*** Além disso, *ASSUMA QUE NENHUM JOGADOR POSSUI DUAS OU MAIS CARTAS IDÊNTICAS EM SUA PRÓPRIA MÃO*, ou seja, assuma que todas as 5 cartas na mão de um único jogador são distintas entre si.
f) Crie a função ***valor***, que receberá uma **carta** de entrada e retornará um **inteiro** correspondente ao seu valor. Lembre-se: o valor de uma carta corresponde ao seu número SE A CARTA NÃO É DE COPAS. Cartas de copas valem **duas vezes** o seu número. Por ex.: um "10 de espadas" vale 10 pontos, mas um "7 de copas" vale 14 pontos.
g) Crie a função **valorDaMao**, que recebe um jogador de entrada e retorna o valor total de suas cartas.
h) Crie a função **comparaMao**, que recebe 2 jogadores de entrada e retorna o ID daquele cuja mão tem o maior valor total de cartas, ou 0 se ambos portam valores equivalentes.
i) Crie a função **melhorDaMesa**, que recebe um vetor **não-ordenado** de jogadores e o seu tamanho N e retorna a posição (índice) do vetor onde está o jogador com o maior valor total em mãos.
j) Crie a função **minhasCartas**, que recebe um jogador e imprime todas as suas cartas no formato "(<numero> de <naipe>)". Por ex., um “9 de copas” será impresso “9 de c”. Sua função NÃO DEVE RETORNAR VALORES.
k) Implemente uma função que receba (i) um vetor com todos os jogadores e também (ii) um outro vetor com todas as N cartas reveladas por cada jogador, e forneça o **status da rodada**. Sua função deverá ter a seguinte assinatura (onde N é o tamanho do vetor ‘reveladas’):

void statusRodada(jogador jogadores[], carta reveladas[], int N);

→ *Assuma que índices equivalentes nos 2 vetores se referem a um mesmo jogador. Por ex., o jogador no índice 4 do vetor **jogadores** é exatamente quem revelou a carta no índice 4 do vetor **reveladas**.*

Essa função deve cumprir as 3 seguintes tarefas, exatamente nesta ordem:

1) ordenar o vetor 'reveladas' em ordem crescente de pontos;
2) para cada carta do vetor (agora já ordenado!), imprimir (i) o **nome** do jogador que a revelou e (ii) quantos **pontos** ela vale.
3) Imprimir o **nome** do **vencedor da rodada** e o **total de pontos** que ele ganhou nessa mesma rodada.

l) (1.5 pontos) Do 2o jogador em diante, cada jogador deve levar em conta se a carta que pretende jogar pode de fato ser revelada, pois a mesma não pode valer a mesma quantidade de pontos de nenhuma das cartas já reveladas até o momento. Escreva uma função que recebe uma carta que o jogador da vez pretende revelar e o conjunto de cartas reveladas até o momento já ordenado por pontos, e retorna 1 caso a carta intencionada possa de fato ser jogada ou 0 caso contrário. A assinatura da sua função é:

int possivel(carta intencao, carta jaReveladas[], int t); *//t é o total de cartas já reveladas.*

Nessa função, **a busca deve ser binária.**

Questão 5)

a) Assim como fizemos no Laboratório 9, crie um **tipo de dados** para representar uma **Pessoa,** mas agora, além dos campos **idade** (inteiro) e **peso** (float)**,** inclua também o campo **nome** (string).

Implemente a função **main,** onde um vetor de Pessoas (máximo de 50) deverá ser preenchido pelo usuário. Ordene esse vetor em ordem **decrescente** de **idade** e imprima o vetor ordenado na tela. **Crie essa lógica de ordenação na própria main**.

→ **Agora, modifique o programa como segue:** após a ordenação do vetor, permita que o usuário insira uma idade. A partir daí, **crie uma função** que receba de entrada o vetor ordenado e a idade informada, e então realize uma **busca binária** no vetor de Pessoas, retornando o índice no vetor onde a pessoa possui a referida idade, ou -1 se ninguém for encontrado. No caso de duas ou mais pessoas com idades iguais, basta retornar um único índice. **Na main**, **imprima o nome dessa pessoa.**

CUIDADO: o vetor está em ordem **decrescente.** Sua busca binária leva isso em conta!

b) Altere o item 'a' de modo que o vetor de Pessoas seja agora ordenado em ordem **crescente** de **peso.**

→ Após a ordenação do vetor, permita que o usuário insira um **peso**. A partir daí, **crie uma função** que receba de entrada o vetor ordenado e o peso informado, e então realize uma **busca binária** no vetor de Pessoas, retornando o índice no vetor onde a pessoa possui o referido peso, ou -1 se ninguém for encontrado. No caso de duas ou mais pessoas com pesos iguais, basta retornar um único índice. **Na main**, **imprima o nome dessa pessoa.**

CUIDADO: agora o vetor está em ordem **crescente.** Sua busca binária leva isso em conta!

Questão 6) O determinante de uma matriz 3x3 pode ser obtido pela Regra de Sarrus, abaixo:

$$\det\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = (aei + bfg + cdh) - (afh + bdi + ceg).$$

Escreva um programa em C que, dada uma matriz quadrada 3 x 3 de entrada (do teclado), calcule o seu determinante. Sua solução deve realizar operações que acessem elementos de uma matriz pelos seus respectivos índices ao invés de criar 9 variáveis para a fórmula acima.