Programação de Computadores I (ICP131)
Prof. Ronald Souza – IC/UFRJ

Gabarito da Segunda Prova – 10/07/2023

Questão 1) (1.0 ponto) O que o código abaixo imprimirá? No seu caderno de respostas, **informe claramente** a alternativa que você escolheu dentre as alternativas a seguir.

```
   a) vexame      c) exam          e) maxe
   b) exame       d) emaxe
#include <stdio.h>
void imprimir(char s[], int pos) {
    if (!pos)
       return;
    imprimir(s, pos - 1);
    printf("%c", s[pos]);
}
int main() {
    char palavra[] = "vexame";
    imprimir(palavra, sizeof(palavra) - 1); return 0;
}
```

RESPOSTA: Letra (b), exame
Comentários:
Dentre as respostas **erradas**, as que mais se aproximam da corretude são as alternativas **(a)** e **(c),** para as quais será atribuído 0.5 ponto, pois preservam o sentido correto da impressão, que **não** é "de trás pra frente", **o que expressa entendimento do passo de recursão**.

- A alternativa (c) exclui corretamente a impressão da letra 'v', mas expressa falha em interpretação de string e indexação de vetor;
- A alternativa (a) inicia a recursão da posição correta, mas demonstra falha de entendimento do teste lógico do caso base, imprimindo incorretamente a letra 'v';

Alternativa (d): será considerada a visão do caso base (exclusão da letra 'v' - 0.2 pontos - 20% da questão).
Alternativa (e): zero. Erra interpretação de string / indexação de vetor, passo de recursão e teste lógico do caso base.

`palavra` é uma **string**, e portanto possui caractere nulo (terminador). Assim, `sizeof(palavra)` retorna 7 (como vimos na Aula 10). Sua representação em vetor é, portanto:

```
'v'  'e'  'x'  'a'  'm'  'e'  '\0'
 0    1    2    3    4    5    6
```

"`sizeof(palavra) - 1`" resulta em 7 - 1 = **6**, que é a posição do caractere terminador.

Como vimos ao longo do curso, uma cláusula if realiza um **teste lógico**. Qualquer expressão interna ao if é interpretada como expressão lógica. No C, um inteiro diferente de 0 equivale a **verdadeiro**, e o 0 representa **falso**. Portanto, **`if(!pos)` equivale ao teste `if(pos == 0).`**

Curiosidade: Note que, ainda que a primeira chamada da função na main fosse "`imprimir(palavra, sizeof(palavra) - 2)`" a alternativa correta continuaria sendo a (b)!

Questão 2) (2.0 pontos) Considere um mapa de ligações entre cidades representado em uma matriz M quadrada, de dimensão NxN, onde o conteúdo da célula M(i,j) indica se existe ou não uma estrada que liga a cidade *i* à cidade *j* **nessa direção** (i.e., de *i* para *j*). Quando uma estrada da cidade *i* para a cidade *j* existe, a célula M(i,j) recebe valor 1; caso contrário, M(i,j) recebe valor 0.

Escreva uma função em C que receba como entrada uma matriz M de ligações e sua dimensão N, e retorne o índice da matriz referente à cidade com o **menor** número de ligações que **chegam até ela**. Em caso de empate, basta retornar qualquer um dentre os índices válidos. Considere 1 < N < 100.

RESPOSTA:

```
int menor(int M[N][N], int n) {
    int menor = N, atual = 0, indiceMenor = 0;
    for (int j = 0; j < N; j++) {
        atual = 0;
        for (int i = 0; i < N; i++) //fixada uma coluna j, somamos seus elementos.
        atual += M[i][j];
        if(atual < menor){
            menor = atual;
            indiceMenor = j;
       }
    }
    return indiceMenor;
}
```

Comentários: Cada célula (i,j) da matriz representa se existe uma estrada pela qual pode-se **partir** de uma cidade i e **chegar** à cidade j diretamente, **nessa direção**. As linhas da matriz portanto indicam as cidades de origem (partidas), e as colunas, as cidades de destino (chegadas). Como queremos a cidade com menor número de estradas que nela **chegam, queremos a coluna j cuja soma dos elementos é a menor dentre todas as colunas.**

EXEMPLO MINIMALISTA DE USO (DENTRO DA main):

```
#include <stdio.h>
#define N 3
int menor(int M[N][N], int n);
int main() {
    int estradas[N][N] = { {1,1,0},{0,1,0},{1,1,1} };
    printf("Indice da cidade onde chegam menos estradas = %d", menor(estradas, N));
    return 0;
}
```

Note que a matriz “estradas” corresponde a

1 1 0
0 1 0
1 1 1

No exemplo acima, dentre as cidades 0, 1 e 2, será impresso o índice 2 (pois a soma dos elementos dessa coluna é igual a 1, valor menor que o mesmo somatório para qualquer outra coluna).

Questão 3) (1.5 ponto) Considere a função definida abaixo. Essa função recebe dois vetores como entrada e deveria imprimir na tela os elementos desses dois vetores de forma intercalada, i.e., o primeiro elemento do primeiro vetor, depois o primeiro elemento do segundo vetor e assim sucessivamente. Por exemplo, se entrarmos com os vetores vetA = [1, 2, 3, 4, 5, 6, 7, 8, 9] e vetB = [0, 0, 0, 0, 0, 0, 0, 0, 0], a sequência de saída será: 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0

```
void intercala(int vetA[], int tamA, int vetB[], int tamB) {
    int a=0, b=0; //indices dos vetores
    for(int i=0; i<(tamA+tamB); i++)
        if(i%2 == 0) { printf("%d ", vetA[a]); a++;} //prox elemento vem de vetA
        else         { printf("%d ", vetB[b]); b++;} //prox elemento vem de vetB
}
```

Ocorreu, entretanto, que ao ser chamada com os vetores de entrada: vetA = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] e vetB = [2, 2, 2, 2, 2], a sequência impressa na tela foi: 1 2 1 2 1 2 1 2 1 2 1 0 1 4196381 1.

Explique o que está errado na função acima e quais seriam as possíveis soluções.

RESPOSTA:

O teste do "for" não leva em conta que os vetores podem ter tamanhos diferentes. (0.75 pontos)

Para corrigir, o teste do "for" deve a cada iteração verificar se os índices de ambos os vetores são válidos, e manter o rastreio desses índices. Ao fim desse laço, há duas possibilidades: se apenas a impressão do intervalo intercalado interessa, terminar; se todos os elementos dos dois vetores devem ser impressos, adicionar outro comando de repetição que imprima os elementos restantes do maior vetor. (0.75 pontos)

Questão 4 (2.0 pontos) A série infinita mostrada abaixo pode ser usada para estimar o valor de **log(1 + *x*),** sendo **−1 < *x* <= 1**. Usando essa série, escreva uma **função recursiva** em C para estimar **log(1 + *x*)**.

Tanto o valor de ***x*** quanto o número ***n*** de **elementos da série** deverão ser passados como argumentos para a função. Considere que ***x*** e ***n*** de entrada já são valores defendidos, e portanto válidos.
É permitido usar a função pow() da biblioteca <math.h>.

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} ...$$

RESPOSTA:

```
double log1maisX(double x, int n) { //Assume-se x e n já defendidos.
    return (n == 1) ? x : -pow(-x, n) / n + log1maisX(x, n - 1);
}
```

Comentários: Embora aplicar a alternância de sinal diretamente na chamada recursiva **seja incorreto** – pois o sinal mais externo é aplicado sobre **todas as chamadas subsequentes** – uma resposta semelhante à resposta abaixo será 90% considerada (1.8 pontos):

```
double log1maisX(double x, int n) {
    if (n == 1) return x;
    if (n % 2)  return pow(x,n)/n - log1maisX(x, n - 1);
    return pow(x,n)/n + log1maisX(x, n - 1);
}
```

Questão 5) (3.5 pontos) Considere um jogo de cartas para N jogadores, N > 1, 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 informar quantos pontos cada jogador revelou a cada rodada.

a) (2.0 pontos) Implemente uma função em C que receba um vetor com todas as N Cartas reveladas, e forneça o status da rodada. Especificamente, sua função deverá:

- utilizar a **estrutura de dados** `Carta` **já fornecida** mais abaixo.
- ter a seguinte assinatura (onde N é o tamanho do vetor '`reveladas`'):

```
void statusRodada(Carta reveladas[], int N);
```

- **ordenar** o vetor '`reveladas`' em ordem crescente de **pontos**
- para cada carta do vetor (agora já ordenado!), imprimir o **nome do jogador** que a revelou e **quantos pontos ela vale**.

DICA: Uma função auxiliar que, dada uma carta, retorna quantos pontos ela vale será bem útil!

```
typedef struct {
    char naipe;   //'o','e','c' ou 'p'(ouros, espadas, copas ou paus, respec.).
    int numero;   //Número da carta, entre 2 e 10.
    char jogador[100];  //Nome do jogador que a revelou;
} Carta;
```

b) (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.

(RESPOSTAS NA PRÓXIMA PÁGINA)

RESPOSTA - Questão 5, item (a):

```
//Função auxiliar que retorna os pontos de uma carta. Usada nos itens (a) e (b):
int pontos(Carta ct) {
    return (ct.naipe == 'c') ? 2 * ct.numero : ct.numero;
}

void statusRodada(Carta reveladas[], int n) { //Bubble sort + impressão
    const int FALSO = 0, VERDADEIRO = 1;
    int ordenado = FALSO;
    int fim = n;
    while (!ordenado) {
    ordenado = VERDADEIRO;
        for (int i = 0; i < fim - 1; i++) {
            if (pontos(reveladas[i]) > pontos(reveladas[i + 1])) {
                Carta aux = reveladas[i];
                reveladas[i] = reveladas[i + 1];
                reveladas[i + 1] = aux;
                ordenado = FALSO;
             }
        }
        fim--;
    }
    for (int i = 0; i < n; i++) //Impressão
        printf("%s: %d\n", reveladas[i].jogador, pontos(reveladas[i]));
}
```

RESPOSTA - Questão 5, item (b):

```
int possivel(Carta intencao, Carta jaReveladas[], int t) {
    int ini = 0, fim = t - 1, meio;
    while (ini <= fim) { //Busca binária pois vetor já está ordenado (por pontos)
       meio = (ini + fim) / 2;
       if (pontos(jaReveladas[meio]) == pontos(intencao))
             return 0; //Valor equivalente já foi revelado. Carta inválida.
       if (pontos(jaReveladas[meio]) < pontos(intencao))
             ini = meio + 1;
       else
             fim = meio - 1;
    }
    return 1; //Carta pode ser revelada!
}
```

Comentário: A aplicação de **busca sequencial** no item (b) ao invés de **busca binária** é **incorreta**, pois o vetor **está ordenado (por pontos)**. Nesse caso, serão considerados 2/3 da questão (1.0 ponto) se o critério da busca (por **pontos** da carta) e o valor de retorno (0 ou 1) estiverem corretos.