Aula prática - Semana 6 - ERE

Nesta lista, as questões 1 e 2 constituem o conjunto de atividades avaliativas que serão utilizadas para contabilizar notas de atividades práticas. As demais questões são desafios que não contabilizam nota, mas cuja realização é desejável para exercitar lógica. As questões 1 e 2 possuem exemplos de execução associados. Note que são apenas exemplos, uma vez que vocês podem propor suas próprias formas de entrada e saída de dados. No entanto, é desejado que se assemelhe ao padrão apresentado no exemplo.
Os seguintes pontos serão considerados na avaliação:

- Atender os requisitos do enunciado.
- Indentação e organização de código.
- Utilização dos tipos corretos de variáveis.
- Utilização correta de constantes nomeadas quando necessário.
- Observação das boas práticas de programação vistas em aula.
- Realização de validação quando a questão requisitar.

Dica: Em tempo de desenvolvimento, você pode começar declarando o vetor já inicializado com valores de teste, e se concentrar no processamento de informações. Isso facilita os testes do algoritmo. Então, uma vez que o programa estiver ok, substituir os vetores inicializados na declaração pelas operações de leitura dos valores adequadas.

1. Você deve construir um programa que lê N valores reais e os armazena em um vetor de N posições. Considere N como 10 (como constante nomeada, ou seja, **#define N 10**), mas o programa deve funcionar para qualquer valor de N (caso eu altere o valor da constante, o programa deve funcionar normalmente). Alternativamente, você pode preencher este vetor aleatoriamente (pode definir um intervalo de início e fim dos números se quiser). A seguir, o programa deve fazer o seguinte (todas as operações são realizadas sobre o mesmo vetor):

- Exibir todos os valores do vetor na mesma linha separados por tabulação (‘\t’).
- Pedir para o usuário ler um novo valor e inserir este valor na primeira posição do vetor, deslocando todos os elementos para as posições subsequentes (deslocamento para a direita). O último elemento é perdido. Exibir o vetor resultante novamente.
- Alterar todos os elementos que estão em posições ímpares do vetor pelo índice da posição. Ou seja, o elemento que está na posição 1 é substituído por 1, e assim sucessivamente. Exibir o vetor resultante novamente.
- Calcular e exibir a média dos valores da última versão do vetor.
- Identificar e exibir o valor da última versão do vetor que está mais próximo da média calculada anteriormente. Caso mais de um elemento satisfaça essa condição, exibir apenas um deles (mas fique à vontade para exibir todos).

Exemplo de execução:
Supondo o seguinte vetor preenchido:

4.0	2.0	-18.0	20.0	100.0	7.0	50.0	12.0	0.0	32.0

Exibindo o vetor original:
4.0 2.0 -18.0 20.0 100.0 7.0 50.0 12.0 0.0 32.0
Informe um valor: -3.0
Exibindo o novo vetor com deslocamento:
-3.0 4.0 2.0 -18.0 20.0 100.0 7.0 50.0 12.0 0.0
Exibindo o novo vetor com substituições:
-3.0 1 2.0 3 20.0 5 7.0 7 12.0 9
A média dos elementos do vetor é: 6.3
O elemento mais próximo da média é 7.0

2.Você deve desenvolver um sistema para gerenciar contas bancárias. Suponha que o banco tenha QTD contas. Assuma que QTD seja 3 por conveniência, embora o programa deva funcionar para qualquer valor de QTD (utilize constante nomeada, ou seja **#define QTD 3**). Os saldos das contas são representados por vetores de números reais. O código de cada conta é a posição do vetor. O programa deve inicialmente ler os saldos iniciais das contas bancárias. A seguir, ele deve continuamente ler uma das opções e efetuar a operação correspondente:

- 0-Sair do sistema: O programa para de ler opções e exibe a mensagem "sistema finalizado".
- 1-Saldo: O programa deve ler o código da conta a informar o saldo da conta. Caso o código seja inválido, deve-se informar isso e voltar a exibir o menu de opções.
- 2-Depósito: O programa deve ler o código da conta e um valor maior que zero para depositar, efetuar a operação, atualizar o saldo e exibir o novo saldo. Caso o código ou o valor sejam inválidos deve-se exibir uma mensagem informando isso e voltar a exibir o menu de opções.
- 3-Saque: O programa deve ler o código da conta e um valor maior que zero para sacar, efetuar a operação, atualizar o saldo e exibir o novo saldo. Caso o saldo seja insuficiente, deve-se informar isso sem efetivar o saque, e voltar a exibir o menu de opções. Caso o código ou o valor sejam inválidos deve-se exibir uma mensagem informando isso e voltar a exibir o menu de opções.
- 4-Transferência: O programa deve ler o código da conta de origem, o código da conta de destino e um valor maior que zero para transferir, efetuar a operação, atualizar os saldos de ambas as contas e exibir o novo saldo da conta de origem. Caso o saldo da conta de origem seja insuficiente, deve-se informar isso, sem efetivar a transferência, e voltar a exibir o menu de opções. Caso os códigos ou o valor sejam inválidos deve-se exibir uma mensagem informando isso e voltar a exibir o menu de opções.

Exemplo de execução:
Supondo o seguinte vetor de contas preenchido depois de interagir com o usuário:

5000.00	2000.00	100.00

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 1
Informe o código da conta: 2
O saldo atual da conta 2 é de R$ 100.00

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 2
Informe o código da conta: 4
Código inválido!

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 2
Informe o código da conta: 1
Informe o valor a ser depositado: 0
Valor inválido!

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 2
Informe o código da conta: 1
Informe o valor a ser depositado: 200.00
Operação efetuada com sucesso.
O saldo atual da conta 1 é de 2200.00

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 3
Informe o código da conta: 0
Informe o valor a ser sacado: 2000.00
Operação efetuada com sucesso.
O saldo atual da conta 0 é de R$ 3000.00

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 4
Informe o código da conta de origem: 0
Informe o código da conta de destino: 2
Informe o valor a ser transferido: 2500.00
Operação efetuada com sucesso.
O saldo atual da conta 0 é de R$ 500.00

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 3
Informe o código da conta: 0
Informe o valor a ser sacado: 2000.00
Saldo insuficiente!

Informe a opção desejada (0-sair, 1-Saldo, 2-depósito, 3-Saque, 4-Transferência): 0
Sistema finalizado

3. (Desafio 1) Faça um simulador de corridas de cavalos. O programa deve simular a corrida de N cavalos. Considere N como 5 por conveniência, embora o programa deva funcionar para qualquer N. O programa deve ler dois valores: dist, a distância da pista; e distMax, a distância máxima do passo. A cada passo, para cada cavalo, deve-se sortear um valor **aleatório** entre 0 e distMax, que vai indicar o quanto o cavalo andou naquele passo. A corrida ocorre até que algum cavalo tenha alcançado o fim da pista. A cada passo da corrida, exiba na tela qual foi a distância já percorrida por cada cavalo. Quando isso ocorrer, o cavalo vencedor e aquele que alcançou a maior distância no fim do processo (note que no último passo, mais de um cavalo pode ultrapassar a linha de chegada). Exiba o número do cavalo vencedor.

4. (Desafio 2) Faça um programa que lê um número de N dígitos (considere #define N 5), armazenando cada dígito em uma posição do vetor. O programa deve descobrir se o número informado é “runaround”. Um número é runaround se ele satisfizer as seguintes condições:

- Cada dígito informa em que posição o próximo dígito ocorre. Por exemplo, no caso do número 38241, o primeiro dígito é 3, o que quer dizer que o próximo dígito a ser visitado está a 3 dígitos à direita dele. Ou seja, o próximo dígito é o dígito 4, na posição 4.
- Quando a contagem chega ao último dígito da direita, ela continua no primeiro dígito da esquerda. Por exemplo, continuando a sequência anterior, do dígito 4, o próximo dígito seria o dígito 2, na posição 3. Porque a seguinte sequência de visitas deveria ser feita: 1-3-8-2.
- A sequência deve retornar ao dígito inicial, depois de todos os dígitos terem sido visitados uma vez.

Por exemplo, o número 81362 é runaround, por que: O primeiro dígito visitado é 8, o segundo visitado é 6, o terceiro é 2, o quarto é 1, o quinto é 3 e o sexto é 8, voltando ao número inicial, visitando todos os números.

5. (Desafio 3) Eratóstenes foi o bibliotecário da biblioteca de Alexandria, nascido em 276 a.c. Ele é conhecido por ter sido um dos primeiros a calcular uma aproximação razoável para a circunferência da Terra (Sim! Ele já sabia que a Terra não era plana!). Ele também desenvolveu um método para calcular os primeiros N números primos, conhecido como “Crivo de Eratóstenes”. O objetivo é desenvolver um programa inspirado no Crivo de Eratośtenes para preencher um vetor com os primeiros N números primos. O algoritmo proposto reconhece que os números primos são análogos a “átomos dos números inteiros”, uma vez que qualquer número não primo é constituído por uma multiplicação de números primos. Neste sentido, para descobrir se um dado número x é primo, não é necessário verificar todo número y entre 1 e x e testar se x é divisível por y. Em vez disso, para testar se x é primo, basta verificar se ele não é divisível por nenhum primo menor que x. O algoritmo proposto mantém um vetor de N posições que armazena números primos e que vai sendo preenchido sequencialmente e segue a seguinte lógica:

    - O algoritmo inicia com x valendo 2, que é o primeiro número maior que 1.
    - Para cada valor de x, testa-se x com todos os números primos que já estão armazenados no vetor, verificando se x é divisível por algum deles. Caso x não seja divisível por nenhum dos números primos do vetor (ou caso o vetor ainda esteja vazio), isso significa que x é um número primo e, portanto, deve ser adicionado ao vetor.
    - Repetimos o processo para x+1, e assim sucessivamente, até preencher o vetor.

No fim do programa, o vetor de números primos deve ser exibido para o usuário. Considere N como 50 por conveniência, embora o programa deva funcionar com qualquer valor de N.

6. (Desafio 4) Faça um programa que lê 11 dígitos (cada dígito é um número entre 0 e 9) e os armazena em um vetor de 11 posições. A seguir, o programa deve verificar se este conjunto de dígitos é um CPF válido. Verifique este site para compreender como são calculados os dígitos verificadores do CPF. Com base neste conhecimento, desenvolva o programa do enunciado.

7. (Desafio 5) Uma forma muito simples de criptografia é a cifra de substituição. Neste método, criamos uma tabela de correspondência que indica por qual caractere, cada caractere do alfabeto é substituída em um dado texto de entrada. Faça um programa que lê duas strings: a primeira representando quais letras serão trocadas e a segunda representando por quais letras as letras da primeira string serão trocadas. Note que a correspondência entre o caractere que deve ser trocado e o caractere substituto é dada pela posição dos caracteres nas duas strings de entrada. Assuma que essas strings só contêm caracteres alfabéticos minúsculos (sem caracteres de pontuação, espaços ou caracteres especiais). Assuma também que não há caracteres repetidos em cada string e que cada string tem, no máximo, 26 caracteres (neste caso, indicando que todas as letras do alfabeto serão trocadas por alguma outra letra). A seguir, o programa deve ler uma frase de até 100 caracteres que será cifrada. O programa deve gerar uma nova frase, substituindo as letras de acordo com a correspondência das duas strings iniciais. Note que se o texto a ser cifrado contiver caracteres que não estão na primeira string, esses caracteres devem aparecer no texto cifrado de forma inalterada (sem substituição).

Exemplo de execução:
Informe os caracteres que devem ser trocados: “helowrd”
Informe os caracteres substitutos: “abcdefg”
Informe a frase que deve ser cifrada: “hello world!”
Frase cifrada:”abccd edfcg!”