Introdução ao Processamento de Dados (IPD)
2025.1 – Turma
Segunda Lista de Exercícios de Programação

1. Faça um programa para ler 50 valores de temperaturas em graus Celsius. Transformar essas temperaturas em Farenheit e imprimir a média das temperaturas em Celsius e Farenheit e quantas temperaturas ficaram acima da média em Farenheit.

2. Leia um vetor de 40 posições e conte quantos elementos pares se encontram no vetor.

3. Leia uma frase e imprima o total de vogais, o total de brancos e o total do resto.

4. Ler um vetor de números inteiros de 30 posições. Depois, ler um número inteiro X, imprimir quantas vezes o número X aparece no vetor.

5. Leia um vetor de 16 posições e troque as 8 primeiras posições pelas 8 últimas posições. Imprima o vetor original e o vetor trocado.

6. Um palíndromo é uma cadeia que pode ser lida de frente para trás e de trás para frente. Exemplos: ‘SOMOS’, ‘1234321’. Implemente a função *palindromo(palavra)*. O parâmetro palavra é uma *string*. A função deverá retornar *True* se for um palíndromo e *False* caso contrário.

7. Um anagrama (do grego ana = "voltar" ou "repetir" + graphein = "escrever") é uma espécie de jogo de palavras, resultando do rearranjo das letras de uma palavra ou frase para produzir outras palavras, utilizando todas as letras originais exatamente uma vez. Um exemplo conhecido é o nome da personagem Iracema, claro anagrama de América, no romance de José de Alencar. Implemente a função *anagrama(frase1, frase2)*. Os parâmetros *frase1* e *frase2* são *strings*. A função deverá retornar *True* se forem anagramas e *False* caso contrário. Despreze acentuação (por exemplo: ã = a e ç = c) e os espaços não devem ser computados para efeitos do anagrama.

8. Crie a função *mat_transposta(matriz)*. A função deve receber uma matriz genérica bidimensional, de qualquer tamanho (não necessariamente quadrada) e retornar a matriz transposta, sem alterar a matriz original.

9. Crie a função *mat_maior_10(matriz)*. A função deve receber uma matriz genérica, de qualquer tamanho (não necessariamente quadrada) e retornar a quantidade de elementos da matriz maiores do que dez.

10. Crie a função *multiplica_matriz(mat1, mat2)* que deve retornar o produto de duas matrizes bidimensionais genéricas, sem alterar as matrizes originais. A função deve imprimir uma mensagem de erro e retornar um vetor vazio ([]) caso não seja possível realizar o produto das duas matrizes.

11. Faça um programa que calcule o valor de π pela soma dos n primeiros termos da série abaixo:

$$\sqrt{12 * (1 - \frac{1}{4} + \frac{1}{9} - \frac{1}{16} + \frac{1}{25} - \frac{1}{36} \ldots}$$

12. Escreva uma função para condensar os elementos de uma lista ordenada L, que contém inteiros repetidos. Por exemplo, para L = [3, 3, 3, 3, 7, 7, 13, 13, 23], a função retorna ['3^4', '7^2', '13^2', '23'] (repare que são *strings*). Note-se que no caso de um número aparecer uma única vez, não deve haver expoente unitário.

13. Crie um algoritmo que leia um número N e imprima os N primeiros números primos. O seu programa deve fazer o MÍNIMO de interações possíveis e obrigatoriamente usar uma função para calcular se o número é primo.

14. Crie um algoritmo de caixa eletrônico que lê a quantidade de dinheiro a ser sacado e imprime a menor quantidade de notas a ser dada ao usuário. A sua máquina de saque eletrônico é carregada com dez notas de 100, 50, 20, 10, 5 e 1 cada. Imprimir também a quantidade de cada nota a ser dada ao usuário. Neste exercício, você tem uma quantidade finita de cada tipo de nota. Exemplo: 98 = uma nota de 50, duas notas de 20, uma nota de 5, e três notas de 1. Perceba que a cada saque há uma quantidade menor de cada tipo de notas. Outra situação é quando não é possível sacar o dinheiro com a quantidade de notas existentes. Neste caso, o programa deve cancelar a operação e informar o usuário que não é possível realizar o saque.

15. Crie um algoritmo que leia um número (com qualquer número de dígitos) em uma base numérica de ordem < 10 e calcule o número correspondente na base decimal. O número da ordem da base (e.g., 2 para binária, 3 para ternária, 8 para octal, etc.) deve também ser informado pelo usuário.

16. Crie um algoritmo que leia um número decimal (com qualquer número de dígitos) e o converta para a base hexadecimal. A resposta deve ser dada em *string*.

17. Faça um programa que preencha uma matriz 3x3 com valores aleatórios e calcule o seu determinante. Uma vez a matriz lida, a parte que faz o cálculo do determinante não se pode ter mais de 4 linhas e não pode haver mais de três células especificadas por linha (em outras palavras, você não pode colocar em uma única linha a fórmula do determinante).

18. Implemente a função *Cramer(matriz, vetor)* que calcule as raízes de um sistema de 3 equações e 3 incógnitas pela Regra de Cramer e retorna a solução na forma de um vetor. Se não houver solução para o sistema, deve imprimir uma mensagem de erro e retornar um vetor vazio ([]). Sua função pode (e deve) chamar a função *det(matriz)* que retorna o valor do determinante da matriz.

19. Leia um arquivo *texto.txt* e conte a quantas vezes cada palavra ocorre no texto, criando dois vetores distintos ou utilizando a estrutura de dicionários. Você tem ainda que considerar que o texto possui pontuação e números, que não deverão fazer parte da contagem.

20. Crie um vetor com 100 elementos aleatórios. Após isso faça a ordenação dos mesmos, SEM UTILIZAR O MÉTODO *sort()*.

21. Leia um arquivo *entrada.txt* e troque todas as letras minúsculas por maiúsculas e vice-versa. Grave o resultado em um arquivo *saida.txt*.

22. Leia um arquivo *entrada.txt* e conte quantas vezes cada letra do alfabeto (independente de ser maiúscula ou minúscula) aparece. Grave o resultado em um arquivo *saida.txt*, com a letra e o número de ocorrências. O arquivo saída deverá ter um formato parecido com:
A 13
B 28
Z 1

23. Implemente a função *area_triangulo(matriz)*. O parâmetro matriz é uma matriz 3x2 que contém as coordenadas x e y de pontos do triângulo. Utilize fórmula de Heron para calcular a área.

$$\sqrt{s.(s-a).(s-b).(s-c)}$$

Onde *s* é o semiperímetro do triângulo e *a*, *b* e *c* são os comprimento dos lados do triângulo.

24. Implemente a função *area_poligono(matriz)*. Esta função calculará a área de um polígono convexo cujos vértices estão na matriz (N x 2 – N indeterminado). Dicas: utilize a função *area_triangulo(mat)* que calcula a área de um triângulo. Em um quadrilátero com vértices X1, X2, X3 e X4, a área do mesmo consiste da soma dos triângulos com vértices (X1, X2 e X3) e (X1, X3 e X4).

25. Escreva um programa para converter números inteiros, menores do que 4000 e escritos em algarismos arábicos, para romanos. Observação: evite escrever mais do que nove "*if*"s.
Dicas:
- A ideia é usar um comando *while* para analisar cada casa decimal e gerar os caracteres romanos diferentemente para cada iteração.
- Use uma *string* ou um dicionário para armazenar as letras correspondentes a cada casa decimal.
Exemplo: 1666 corresponde a string "MDCLXVI", onde:
as letras nas posições 5 e 6 correspondem às unidades (VI),
as letras nas posições 3 e 4 correspondem às dezenas (LX),
as letras nas posições 1 e 2 correspondem às centenas (DC),
a letra na posição 0 corresponde aos milhares (M).

26. Escreva um programa para converter números escritos em algarismos romanos (menores que 4000) para arábicos.

27. Escreva uma função *fat_primo(num)* que decompõe *num* em fatores primos e retorna um vetor com a decomposição.
Exemplo: 564757684810891538218830 27 = ['3^33', '7^4', '11^4', '17^2']

28. Considere alguma sequência de elementos (difere de um mero conjunto de elementos por definir uma ordem entre os seus membros). Enumere, então, todas as subsequências não contínuas de uma dada sequência.
Dicas:
- Uma subsequência sempre contém algum subconjunto de elementos da sequência, na mesma ordem.
- Uma subsequência contínua é aquela em que nenhum elemento está faltando, entre o primeiro e o último elemento da subsequência.
Exemplo: (1,2,3,4) → [[1, 2, 4], [1, 3, 4], [1, 3], [1, 4], [2, 4]]