Prova

Fundamentos de Programação

1. Implemente uma função que calcule as raízes de uma equação do segundo grau, do tipo $(ax^2 + bx + c)$, Essa função deve obedecer ao protótipo:

   ```
   int raizes(float a, float b, float c, float* x1, float*x2);
   ```

   Essa função deve ter como valor de retorno o número de raízes reais e distintas da equação. Se existirem raízes reais, seus valores devem ser armazenados nas variáveis apontadas por `x1` e `x2`.

2. Implemente uma função que calcule a área da superfície e o volume de uma esfera de raio r. Essa função deve obedecer ao protótipo:

   ```
   void calc_esfera(float r, float* area, float* volume);
   ```

   A área da superfície e o volume são dados, respectivamente, por $4r^2$ e $4r^3/3$.

3. Implemente uma função que receba uma string como parâmetro e retorne como resultado o número de vogais nessa string. Essa função deve obedecer ao protótipo:

   ```
   int conta_vogais(char* str);
   ```

4. Implemente uma função que receba uma string e um caractere como parâmetro e retorne como resultado o número ocorrências desse caractere na string. Essa função deve obedecer ao protótipo:

   ```
   int conta_char(char* str, char c);
   ```

5. Implemente uma função que receba uma string como parâmetro e altere nela as ocorrências de caracteres maiúsculos para minúsculos. Essa função deve obedecer ao protótipo:

   ```
   void minusculo(char* str);
   ```

6. Faça uma função que recebe um número n por parâmetro e imprime os valores entre 2 e n, que são divisores de n.

7. Escreva um programa que lê um número n, e então imprime o menor número primo que é maior ou igual a n, e imprime o maior primo que é menor ou igual a n.

8. Faça um programa que leia um número n e imprima n linhas na tela com o seguinte formato (exemplo se $n = 6$):

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

9. Faça um programa que leia um número n e imprima n linhas na tela com o seguinte formato (exemplo se $n = 6$):

```
+ * * * * *
* + * * * *
* * + * * *
* * * + * *
* * * * + *
* * * * * +
```

10. Dizemos que um número natural é triangular se ele é produto de três números naturais consecutivos. Exemplo: 120 é triangular, pois 4 * 5 * 6 = 120. Dado um inteiro não-negativo n, verificar se n é triangular.

11. Faça um programa que lê um número n e imprime os valores entre 2 e n, que são divisores de n.

12. Escreva um programa que lê dois números inteiros x e y e determina se eles são ou não coprimos. Dois números a e b são coprimos se não há um divisor $d > 1$ que seja comum a ambos. Por exemplo, 15 e 8 são coprimos pois os divisores de 8, que são 2, 4 e 8, não são divisores de 15.

13. Escreva um programa que leia um número inteiro n fornecido pelo usuário e imprima um "quadrado" de n linhas e n colunas onde na linha i e coluna j seja impresso o valor 1 caso i e j sejam coprimos e 0 caso contrário. Abaixo temos um exemplo para n = 9.

```
   1  2  3  4  5  6  7  8  9
1  1  1  1  1  1  1  1  1  1
2  1  0  1  0  1  0  1  0  1
3  1  1  0  1  1  0  1  1  0
4  1  0  1  0  1  0  1  0  1
5  1  1  1  1  0  1  1  1  1
6  1  0  0  0  1  0  1  0  0
7  1  1  1  1  1  1  0  1  1
8  1  0  1  0  1  0  1  0  1
9  1  1  0  1  1  0  1  1  0
```

Os números de 1 até 9 na primeira coluna e primeira linha acima foram colocados apenas para ilustração. A saída do seu programa para $n = 9$ deve ser apenas:

```
1  1  1  1  1  1  1  1  1
1  0  1  0  1  0  1  0  1
1  1  0  1  1  0  1  1  0
1  0  1  0  1  0  1  0  1
1  1  1  1  0  1  1  1  1
1  0  0  0  1  0  1  0  0
1  1  1  1  1  1  0  1  1
1  0  1  0  1  0  1  0  1
1  1  0  1  1  0  1  1  0
```

14. Escreva uma função que computa a potência a^b para valores a (real) e b (inteiro) passados por parâmetro - não a função `pow`. Use a função anterior e crie um programa que imprima todas as potências: $2^0, 2^1, ..., 2^{10}, 3^0, ..., 3^{10}, ..., 10^{10}$.

15. Escreva uma `função` que recebe um vetor de inteiros e seu tamanho como parâmetros, e devolve a soma dos números primos deste vetor.

16. Escreva uma função que recebe uma string como parâmetro e retorna a quantidade de caracteres da string. **Nota: não use nenhuma função da biblioteca string.h**.

17. Implemente uma função que recebe duas strings como parâmetro e retorna uma **nova** string formada pela concatenação das duas recebidas. **Nota: não use nenhuma função da biblioteca string.h**.

18. Implemente uma função que recebe duas strings e um inteiro n como parâmetro e retorna uma **nova** string formada pela concatenação da primeira string com os primeiros n caracteres da segunda string **Nota: não use nenhuma função da biblioteca string.h**.

19. Implemente uma função que recebe duas strings como parâmetro e retorna 0 se as strings forem iguais e 1 caso contrário. **Nota: não use nenhuma função da biblioteca string.h**.

20. Implemente uma função que recebe uma string como parâmetro e retorna 0 se a string for palíndroma e 1 caso contrário.