NOTA: _____

ACADÊMICO:	
CURSO: ANÁLISE E DESEN. DE SISTEMAS (ADS)	**SEMESTRE: 2º NOTURNO**
PROFESSOR: JEAN ZAHN	
DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO	

Observações:

i. **1° Lista correspondente a atividade da N1.**
ii. **Entregar impresso, sem folhas adicionais.**
iii. **O preenchimento deverá ser manuscrito.**
iv. **A data limite da entrega é a aula que antecede a Avaliação N1.**

LISTA DE EXERCÍCIOS 5
LISTA FINAL

1. **Calcular a soma dos 100 primeiros nº naturais.**
2. **Calcular os divisores de um nº qualquer.**
3. **Para calcular o fatorial de um número qualquer.**
4. **Imprimir o menor inteiro positivo x cujo quadrado é superior a um valor L dado.**
5. **Imprimir a tabuada de qualquer número n.**
6. **Ler um número e escreva se ele "é primo" ou "não é primo".**
7. **A série de Fibonacci é uma sequência de termos que tem como os 2 primeiros termos, respectivamente, os números 0 e 1. A partir daí, os demais termos são formados seguindo uma certa regra. A série de Fibonacci pode ser vista a seguir:**

0 1 1 2 3 5 8 13 21...

Descubra a regra que gera a sequência da série de Fibonacci e escreva um algoritmo que gere os n (solicitados pelo usuário) primeiros termos desta série e calcula e escreve a soma destes termos.

8. **Gerar 20 números de 1000 a 1999 e escrever aqueles que divididos por 11 dão um resto igual a 5.**
9. **Imprima a tabela ASCII (letra e código decimal correspondente).**
10. **Modifique o programa da média (questão 1 da segunda lista) para que ao final pergunte ao usuário se deseja calcular a média de outro aluno. O programa só será finalizado quando o usuário não desejar mais calcular nenhuma média.**
11. **Gere um número aleatório inteiro (utilize a função rand()) entre 0 e 100 e solicite um número ao usuário. O objetivo é que o usuário acerte o número gerado. Se o número digitado for menor que o gerado, diga "MAIOR", se for maior diga "MENOR", e solicite um número ao usuário novamente. Repita este processo até que o usuário acerte o número gerado. Após isso, informe em quantas tentativas o usuário acertou.**

EXERCÍCIOS DE OO

1. **Crie um arquivo chamado conta.py na pasta OO criada no exercício anterior.**
2. **Crie a classe `Conta` sem nenhum atributo e salve o arquivo.**
3. **Crie uma instância (objeto) da classe Conta e utilize a função type() para verificar o tipo do objeto.**
4. **Além disso, crie alguns atributos e tente acessá-los.**
5. **Abra novamente o arquivo conta.py e escreva o método `__init__()` recebendo os atributos anteriormente definidos por nós que toda conta deve ter (`numero`, `titular`, `saldo` e `limite`):**
6. **Tente criar uma conta sem passar qualquer argumento no construtor.**
7. **Note que o interpretador acusou um erro. O método `__init__()` exige 4 argumentos `'numero'`, `'titular'`, `'saldo'` e `'limite'`.**
8. **Agora vamos seguir o exigido pela classe, pela receita de uma conta.**
9. **O interpretador não acusou nenhum erro. Vamos imprimir o `numero` e titular da conta.**
10. **Crie o método `deposita()` dentro da classe Conta. Esse método deve receber uma referência do próprio objeto e o valor a ser adicionado ao saldo da conta.**
11. **Crie o método `saca()` que recebe como argumento uma referência do próprio objeto e o valor a ser sacado. Esse método subtrairá o valor do saldo da conta.**
12. **Crie o método `extrato()`, que recebe como argumento uma referência do próprio objeto. Esse método imprimirá o saldo da conta:**
13. **Modifique o método `saca()` fazendo retornar um valor que representa se a operação foi ou não bem sucedida. Lembre que não é permitido sacar um valor menor do que o saldo.**
14. **Crie o método `transfere_para()` que recebe como argumento uma referência do próprio objeto, uma Conta destino e o valor a ser transferido. Esse método deve sacar o valor do próprio objeto e depositar na conta destino:**
15. **Abra o Python no terminal, importe o módulo conta, crie duas contas e teste os métodos criados.**
16. **(Opcional) Crie uma classe para representar um cliente do nosso banco que deve ter `nome`, `sobrenome` e `CPF`. Instancie uma Conta e passe um cliente como titular da conta. Modifique o método `extrato()` da classe Conta para imprimir, além do número e o saldo, os dados do cliente. Podemos criar uma Conta sem um Cliente? E um Cliente sem uma Conta?**
17. **(Opcional) Crie uma classe que represente uma data, com dia, mês e ano. Crie um atributo `data_abertura` na classe `Conta`. Crie uma nova conta e faça testes no console do Python.**
18. **(Desafio) Crie uma classe Historico que represente o histórico de uma Conta seguindo o exemplo da apostila. Faça testes no console do Python criando algumas contas, fazendo operações e por último mostrando o histórico de transações de uma Conta. Faz sentido criar um objeto do tipo Historico sem uma Conta?**
19. **Agora, além de funcionar como esperado, nosso código não permite criar uma conta sem os atributos que definimos anteriormente. Discuta com seus colegas e instrutor as vantagens da orientação a objetos até aqui.**