Questão 1: Assinale a alternativa que completa corretamente o texto a seguir:
“Quando um programa é compilado, o binário executável fica armazenado em (A). Quando o usuário executa uma chamada para esse executável, o binário é transferido para (B). Durante sua execução, existem três espaços reservados: (C).”
Os itens A, B e C correspondem a:

a. memória RAM / memória cache / espaço de código, espaço estático e espaço Dinâmico (Pilha e Heap).
b. monitor; lixeira; estático, dinâmico e virtual.
c. disco rígido; memória RAM; espaço de código, espaço estático e espaço Dinâmico (Pilha e Heap)
d. memória virtual; memória local; espaço virtual, espaço estático e espaço dinâmico.

1.1) Gabarito: A

1.2) Justifique as alternativas incorretas (com as suas palavras) em um parágrafo único:

O binário gerado fica armazenado no disco rígido, e na memória RAM ficam armazenados os binário executáveis onde os dados são armazenados de maneira temporária. Se você copiar algo utilizando o CTRL+C, este dado é armazenado na nossa memória RAM, mas ao reiniciar o sistema operacional vai perceber que isso não está mais na área de transferência. Instruções são executadas dentro da cpu, mas lido e escrito na RAM.

Questão 2: Diferencie Pilha e Heap nos seguintes itens: (i) escopo de variáveis; (ii) alocação estática e dinâmica; (iii) binário:
Pilha
Na pilha, são guardados objetos alocados dentro de escopos de funções incluindo variáveis locais das funções, argumentos, endereços das áreas de código sendo executadas antes de outras chamadas de função, retorno de funções.
A alocação de memória ocorre de forma sequencial e, como a posição desses objetos é conhecida durante o tempo de compilação, nós podemos atribuir nomes próprios a esses objetos e acessá-los diretamente. Quando um objeto que é alocado no stack sai de seu respectivo escopo, o objeto é automaticamente deletado. Então você não precisa se preocupar com alocação e desalocação de memória com objetos da pilha mas atenção, o stack tem um tamanho limitado.

Heap
O heap é o local de memória adequado para alocar muitos objetos grandes, pois esta seção do programa é bem maior que a pilha, e seu tamanho é limitado apenas pela memória virtual disponível na sua máquina. Os objetos alocados no heap são todos aqueles alocados usando new ou malloc() (objetos alocados dinamicamente). Como a posição em que esses objetos vão estar durante a execução do programa é desconhecida em tempo de compilação, a única forma de acessá-los é via Pointeros. Deve lembrar-se de controlar a deslocação desses objetos, pois não são destruídos automaticamente.

A Pilha (Stack) é mais rápida pois as variáveis/objetos são criados em tempo de compilação, a pilha não se estende pela memória virtual da máquina (HD) logo em algum

momento um objeto/variável alocado no Heap podem estar armazenado no HD e logo este deverá ser carregado na RAM.

Questão 3: Explique com suas palavras as seguintes definições:

a. Tipos de dados primitivos

INTEIRO: Representa valores numéricos negativo ou positivo sem casa decimal, ou seja, valores inteiros.
REAL: Representa valores numéricos negativos ou positivos com casa decimal, ou seja, valores reais. Também são chamados de pontos flutuantes.
LÓGICO: Representa valores booleanos, assumindo apenas dois estados, VERDADEIRO ou FALSO. Pode ser representado apenas um bit (que aceita apenas 1 ou 0).
TEXTO: Representa uma sequência de um ou mais caracteres, colocamos os valores do tipo TEXTO entre “ " (aspas duplas).

“conjunto de valores (denominado domínio) que uma variável pode assumir ao longo da execução de um programa e do • conjunto de operações que podem ser aplicadas sobre ele.”

b. Tipos de dados estruturados

Um tipo estruturado é um exemplo de estrutura de dados. Tipos estruturados são estruturas de dados já pré definidas na linguagem de programação. O programador pode definir outras estruturas de dados para armazenar as informações que seu programa precisa manipular. Vetores, registros, listas encadeadas, pilhas, filas, árvores, grafos, são exemplos de estruturas de dados típicas utilizadas para armazenar informação em memória principal.

c. Tipos de dados abstratos

Os tipos e estruturas de dados existem para serem usados pelo programa para acessar informações neles armazenadas, por meio de operações apropriadas.
Do ponto de vista do programador, muitas vezes é conveniente pensar nas estruturas de dados em termos das operações que elas suportam, e não da maneira como elas são implementadas.

Questão 4: Tipos de dados abstratos são fundamentais para o desenvolvimento de sistemas complexos, pois eles permitem uma camada de abstração a mais para que o programador possa desenvolver seus códigos com maior facilidade. Justifique essa afirmativa.

Uma estrutura de dados definida dessa forma é chamada de um Tipo Abstrato de Dados (TAD). O conceito de TAD é suportado por algumas linguagens de programação procedimentais. Requer que operações sejam definidas sobre os dados sem estarem atreladas a uma representação específica. Programador que usa um tipo de dado real, integer, array não precisa saber como tais valores são representados internamente.

Questão 5: Na maioria das linguagens de programação, existem duas formas de representar coleções de dados: usando vetores/matrizes ou listas encadeadas.

a. Vetores em (de preferência em C ou Java) são sempre estáticos? Justifique sua resposta.

b. Qual a diferença entre vetores e listas encadeadas?

Uma lista encadeada é uma representação de uma sequência de objetos, todos do mesmo tipo, na memória RAM (= random access memory) do computador. Cada elemento da sequência é armazenado em uma célula da lista: o primeiro elemento na primeira célula, o segundo na segunda, e assim por diante.

Um vetor , ou arranjo (= array), é uma estrutura de dados que armazena uma sequência de objetos, todos do mesmo tipo, em posições consecutivas da memória RAM (= random access memory) do computador.

c. Quando se trabalha com alocação dinâmica, um problema muito comum é a fragmentação de memória durante a execução do programa. O que é fragmentação de memória? Como as listas encadeadas amenizam esse problema quando comparadas a vetores?

Fragmentação é o desperdício de espaço disponível em memória.

Alocação encadeada

• Vantagens: não há fragmentação externa, todo o disco pode ser usado tamanho dos arquivos pode ser mudado facilmente.

• Desvantagens; acesso aleatório é mais demorado, maior fragilidade em caso de problemas.

Questão 6: Crie estruturas abstratas que representam cada uma das seguintes definições: Atenção estas estruturas devem ser definidas apenas no documento, não é necessária a implementação, sigo o exemplo exposto abaixo

a. [EXEMPLO] “Data é o modo pelo qual se define um certo momento no tempo. Normalmente dá-se esse nome a uma forma de designar o número ou o nome de um Dia, Mês ou Ano, muitas vezes de forma conjunta.”

a) Data

```
class DateStruct:
        def __init__(self,day, month, year):
                self.day = day
                self.month = month
                self.year = year
```

b. “No nosso banco, cada cliente é associado por um nome, um CPF, um telefone, um endereço, um CEP e uma conta. Cada conta possui a agência e o número da conta.”

```
class Account :
        def __init__(self, number, agency):
                self.__number = number
                self.__agency.agency

        def setNumber(self, number):
                self.__number = number

        def setAgency(self, agency):
                self.__agency = agency

        def getNumber(self):
                return self.__number

        def getAgency(self):
                return self.__agency
```

```
class clientAccount(Account):
        def __init__(self,name, cpf, phone, address, cep, number, agency ):
                Account.__init__(self, number, agency)
                self.name = name
                self.cpf = cpf
                self.phone = phone
                self.address = address
                self.cep = cep
```

c. “Criamos a rede social MyBook, onde cada usuário possui um identificador único, um e-mail, uma senha, uma lista de seguidores e um mural com mensagens de texto. Cada mensagem de texto pode ter no máximo 128 caracteres.”

```
class myBook :
        def __init__(self, id, email, password):
                self.id = id
                self.email = email
                self.password = password
                self.follow = None
                self.message = None

        def __str__(self):
              return f"Email {self.email}, {self.follow}, {self.message}"

class Followers:
        def __init__(self, nome):
                self.nome = nome
                self.followers = []

        def adiciona_follow(self, follow):
                 follow.follow = self
                self.followers.append(follow)
class  Message:
        def __init__(self):
                self.messages = []

        def adiciona_mesage(self, message):
                 message.message = self
                cont = 128
                if self.message > cont:
                        return “Mensagem não pode ter mais de 128 caracteres”
                else:
                        return self.messages.append(message)
```

Questão 7: Sobre Tipo Abstrato de Dados (TADs), considerando a implementação de Lista de Ingredientes: Atenção estas estruturas devem ser definidas apenas no documento, não é necessária a implementação, sigo o exemplo exposto abaixo.

7.1) Crie um registro/struct em que representa o TAD Ingrediente. Esse registro deve conter três campos: nome, quantidade e medida. Lembrando a forma de como criar TADs (acesse material de apoio) e o exemplo abaixo o qual apresenta a implementação do TAD Data.

7.2) Após, defina, pelo menos, 4 métodos de acordo com o domínio do problema: Lista de Ingredientes. A definições devem seguir o modelo abaixo (nome do método, entrada(s), saída(s), objetivo) :

Método Exemplo acrescentarDias
Entradas: D (data), Dias (inteiro)
Saída: (Data)
Objetivo: método que recebe soma um determinado número de Dias a uma data recebida como parâmetro e retorna o resultado. Caso não seja possível realizar a operação, o método retorna uma data cujo Dia seja -1

Questão 8: Acesse a implementação de Listas Encadeadas (Pasta da Aula 6 – Arquivo: CodigosListaEncadeada.zip).

8.1) Debug o código, realize testes de modo a entender a implementação e responder as perguntas abaixo:

a. Qual o objetivo do código?Implementar uma lista encadeada que insere no inicio.
b. Quantas classes são implementadas? Duas
c. Qual o relacionamento entre elas? Acredito que seja composição
– Herança: significa que A é um B
Por exemplo: Poupança é uma Conta
– Agregação: significa que A tem um B
Num sentido mais fraco que na composição.
Por exemplo, Conta tem um Cliente
– Composição: significa que A tem um B
Num sentido mais forte que na agregação. Por
exemplo, Conta tem um Historico
d. Como os dados ficam armazenados na estrutura proposta?
Numa lista linear simplesmente encadeada cada elemento possui, além do espaço para armazenamento da informação, um espaço para armazenar uma referência da localização na memória onde o próximo elemento da lista (ou o anterior) se encontra.

8.2) Escreva com suas palavras como funciona a implementação apresentada. Utilize diagramas (caixinhas),
fluxogramas, mapa mental (entre outras alternativas) de modo que a explicação fique mais clara / didática
possível.

Questão 9: Sobre o código disponibilizado:

a. Aponte possíveis melhorias no código Exemplo e implemente uma nova versão do código exposto de modo a melhorar os conceitos de: Orientação Objetos, Estrutura de Dados e Linguagem Python.
b. Faça as outras operações necessárias para manipular listas, especialmente o método de REMOVER.

Questão 10: INDO ALÉM: Implemente uma Lista de Datas (Tad definido nas aulas anteriores) utilizando a definição de listas encadeadas vista em aula. Utilize o código exemplo ou aprimorada para esta implementação.