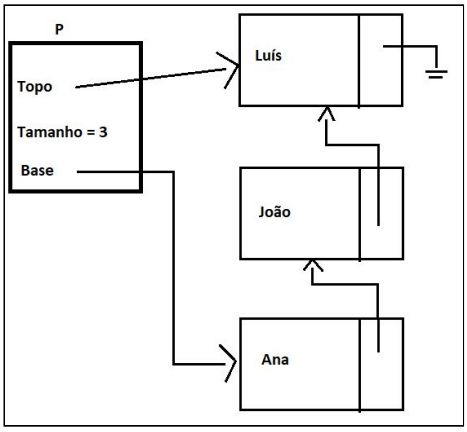
Tarlles Roman Sfredo

Questão 01

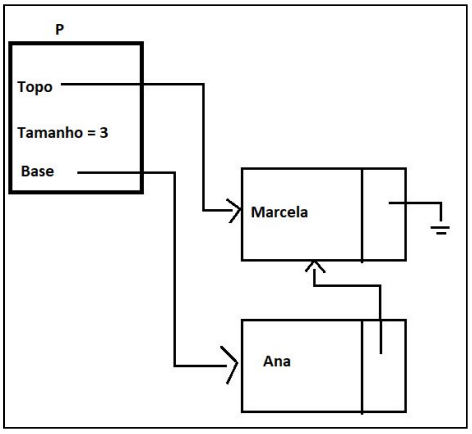
a) Implementação da pilha em C, arquivo “pilha.h”

```c
#ifndef PILHA_H_INCLUDED
#define PILHA_H_INCLUDED

typedef struct {
    char* nome;
} tipo_elemento;

typedef struct nodo{
    tipo_elemento info;
    struct nodo *prox;
}tipo_nodo;

typedef struct {
    tipo_nodo* topo;
    tipo_nodo* base;
    int tamanho;
} tipo_pilha;

typedef tipo_pilha *pilha;

pilha cria_pilha();
int push(pilha,tipo_elemento);
tipo_elemento pop(pilha);
int esta_vazia(pilha);
void termina_pilha(pilha);

#endif // PILHA_H_INCLUDED
```

b)



questao_01_b_sequencia_1.jpg



questao_01_b_sequencia_2.jpg

Tarlles Roman Sfredo

Questão 02

Arquivo questao_02_a_b.c

a)

```c
/** Retorna o menor elemento do vetor de inteiros;
*   retorna 0 caso o tamanho recebido seja menor ou igual a 0
*/
int encontrar_menor(int *vetor, int tamanho){
    int menor, i;

    if (tamanho <= 0) return 0;

    menor = vetor[0];
    for(i = 1; i < tamanho; i++)
        if(vetor[i] < menor) menor = vetor[i];

    return menor;
}
```

b)

```c
/** Como não foi informado qual o tamanho do vetor utilizado,
 *  escolhi o método insertion sort para ordenação crescente de
 *  elementos por ser uma boa escolha em vetores pequenos onde
 *  enfrentaria seu pior caso, tamanho² comparações, apenas
 *  caso o vetor esteja ordenado de forma decrescente.
 * */
void insertion_sort(int *vetor, int tamanho){
  int i, j, aux;

  for(i = 1; i < tamanho; i++){
    j = i;

    while((j != 0) && (vetor[j] < vetor[j - 1])) {
      aux = vetor[j];
      vetor[j] = vetor[j - 1];
      vetor[j - 1] = aux;
      j--;
    }
  }
}
```

Tarlles Roman Sfredo

Questão 03

UML

UML (Linguagem de Modelagem Unificada) é uma linguagem com o objetivo de descrever modelos de sistemas, do mundo real e de software, baseado em conceitos de orientação a objetos. Esses modelos são principalmente diagramáticos, como exemplo temos o diagrama de casos de uso e o diagrama de classes, entre outros.

CLASSE

Classe é um conceito muito comum no paradigma de programação orientada a objetos, elas são a descrição das propriedades (atributos), e/ou comportamentos (métodos), de um conjunto de objetos que serão utilizados em um sistema. Por exemplo, em um sistema para controle de livros em uma biblioteca, seria possível a utilização de uma classe (Pessoa) com atributos (Nome, Sexo, Endereço) e ações (Pegar livro emprestado, Devolver livro emprestado)

OBJETO

Objeto é uma instância da classe citada anteriormente, já que a classe seria uma abstração de um objeto real, cada “objeto” é a representação em si do objeto real. Por exemplo poderíamos ter o objeto da classe Pessoa{ nome: João, sexo: Masculino, endereço: Rua dos bobos, nº 0 }; e esse objeto poderia realizar as ações de empréstimo como qualquer outra pessoa de sua classe.

HERANÇA

Herança é o princípio na programação orientada a objetos que permite que as classes compartilhem atributos e métodos entre si. Herança também é chamado de relacionamento “é um” onde cada um dos objetos classes “filho” também é um objeto da classe “pai”, porém o contrário não é aplicado. Por exemplo, a classe Pessoa poderia ser “pai” das classes Pessoa Física e Pessoa Jurídica, cada Pessoa Física teria os mesmos atributos herdados da classe Pessoa somado ao atributo (CPF) mas não teria o atributo (CNPJ) da classe Pessoa Jurídica

Questão 04

O método encontra quantas posições a partir do Node li está o Objeto v requisitado, a informação retornada na variável inteira representa a quantidade necessária de “saltos” para encontrar tal objeto, ou 0 caso não encontre o objeto e/ou tenha recebido como entrada um valor “null”.

Tarlles Roman Sfredo

Questão 05

Teoria e Prática: aliadas ou adversárias.

Segundo o dicionário, teoria é ¹conjunto de regras ou leis, mais ou menos sistematizadas, aplicadas a uma área específica. ²conhecimento especulativo, metódico e organizado de caráter hipotético e sintético. Enquanto a Prática é ¹execução rotineira (de alguma atividade). ²o que é real, não é teórico; realidade. Seguindo esse prévio conhecimento teórico podemos concluir que prática e teoria são duas adversárias extremamente opostas. Porém qualquer pessoa que já teve que montar algo, como uma cômoda, na prática deve concordar que um pouco de conhecimento teórico facilita o processo, logo podemos concluir que teoria e práticas são boas aliadas. Isso é tão confuso quanto uma amizade de longa data, por que é uma.

Desde os primórdios a humanidade vem aperfeiçoando suas técnicas tanto com a prática quanto com a teoria. A prática ajudou os agricultores a selecionar melhor suas sementes enquanto a teoria os ensinou que plantar em determinadas épocas do ano faziam os frutos se desenvolverem mais facilmente. A teoria fez Santos Dumont acreditar que era possível ao homem voar, mas só a prática o fez realizar esse sonho. A prática faz bons programadores mas a teoria os torna excelentes. Esses exemplos mostram como a teoria e prática são ótimas aliadas e apesar de conseguirem sobreviver sozinhas, quando juntas possuem uma relação de mutualismo com benefícios difíceis de serem alcançados separadamente.

A teoria e prática são aliadas, embora a um observador desatento possa parecer que são adversárias como quem observa a dois velhos amigos jogando cartas em uma mesa.

Questão 06

A aplicação está disponível através da plataforma Heroku, que por ser uma plataforma gratuita não mantém os servidores rodando por 24h, eles são desligados após algum tempo e são reiniciados quando há uma nova requisição, portanto pode ser observada uma lentidão no primeiro acesso. Dito isso, segue o link: https://rerum-trs.herokuapp.com/

O código fonte do front-end da aplicação está disponível em:
https://github.com/TarllesRoman/rerum-frontend

O código fonte do back-end da aplicação está disponível em:
https://github.com/TarllesRoman/rerum-api