IFSP – Instituto Federal de São Paulo (Catanduva)
Tecnólogo em Análise e Desenvolvimento de Sistemas
ED2 – Estrutra de Dados II

Trabalho I

Prof. Flavio Souza - flavio.souza@ifsp.edu.br

INSTRUÇÕES

1. O Trabalho Prático deve ser feito em grupo com 3 componentes.
2. O programa (código) deve ser enviado para o professor por e-mail (flavio.souza@ifsp.edu.br) até a data limite da apresentação (05/05/2022 às 23:59).
3. Trabalhos entregue a posteriori da data limite não serão considerados, e será atribuído a nota ZERO ao grupo.
4. O programa deve ser feito exclusivamente em C.
5. Não são permitidos o uso de bibliotecas e/ou frameworks.
6. Para os códigos que não compilarem e/ou executarem receberão ZERO como nota.
7. O programa deve utilizar uma ou mais estruturas de EDI e/ou ED2.
8. Se não especificado no TEMA a estrutura é de livre escolha do grupo.
9. Se no TEMA especificar LIMITES de tamanho, sinaliza a possibilidade de utilização de estrutura de arranjadores, caso contrário devesse utilizar estruturas de apontadores (gerando a dinamicidade para o TEMA).
10. As manipulações de itens nas estruturas (inserção, remoção, consulta, busca etc.) devem respeitar as condições da estrutura.
11. A organização, estrutura do código são considerados como critérios de avaliação.
12. Funções que apresente o STATUS e as MOVIMENTAÇÔES dos “objetos” do simulador serão valorizados.

TEMAS

Ranking de Futebol (Maycon, Taissa)
Simule, por meio de um algoritmo, a Eleição “The best” da FIFA. A eleição simulada deve ocorrer atrás de votos divididos em quatro grupos de eleitores (Técnicos, Capitães, Jornalistas e Publico Geral). Cada votante tem direito a escolher os três melhores atletas, entre 10, de cada categoria. Ao final da votação ordena-se os atletas por quantidade de votos recebidos, para cada grupo votante, e atribui-se ao atleta 5 pontos para o primeiro, 3 pontos para o segundo colocado e 1 ponto para o terceiro. Os três primeiros, melhores votados são os atletas que obtiverem a maior quantidade de pontos entre os 4 grupos votantes.
A simulação deve conter as funções:

- **votar:** essa função adiciona, em uma lista, um voto recebendo apenas qual o grupo correspondente ao votante e o código do atleta;
- **ordenacaoPorGrupo:** a considerar os votos, essa função deve retornar os 3 atletas mais votados de cada grupo;
- **theBest:** Contabilizando os mais votados de cada grupo, essa função retorna o 3 atletas que receberam mais pontos entre os quatro grupos.

Sistema logístico (Alafhi, Laísa e Susana)
Simule por algoritmo um Sistema Logístico Portuário. A simulação ocorrer pelo carregamento de containers ao navio. Um container é composto por Código e Ponto de Descarregamento (1 à

3 pontos de descarregamento). Para manter a estabilidade do navio, os containers devem ser distribuídos sob três pilhas regidas pela ordem decrescente de Descarregamento.
A simulação deve conter as funções:

- **filaEmbarque:** essa função adiciona, em uma fila, um container recendo apenas o Ponto de Descarregamento e o código do container.
- **carregamento:** a considerar a sequência de containers na fila, o carregamento do container ao navio dever ser simulado distribuídos nas pilhas conforme o critério de carregamento.
- **descarregamento:** conforme a chega aos pontos, os containers do respectivo ponto devem ser descarregados. No descarregamento deve-se verificar a fila de embarque provocando uma reorganização na disposição dos containers.

Drive Thru (Danilo, Henrique e João Victor)
Implemente um algoritmo que simule o funcionamento de um estabelecimento de Drive Thru, na qual os pedidos efetuados no Drive Thru concorrem com os pedidos do Balcão e Delivery. Um pedido é composto por Senha e origemPedido (DriveThru, Balcao, Delivery). A realização dos pedidos deve seguir uma política de priorização. A cada 2 pedidos do Drive Thru (no máximo), um pedido do Balcão deve ser atendido. A cada 3 pedidos do Drive Thru (no máximo) ou 2 pedidos de Balcão (no máximo) 1 pedido do Delivery deve ser atendido.
A simulação deve conter as funções:

- **realizarPedido:** na qual é informado a origem do pedido e a senha deve ser gerada sequencialmente pelo simulador.
- **fazerPedido:** essa função faz o papel da cozinha, sendo assim um pedido da fila deve ser feito ficando disponível para entrega. A cozinha tem um limite de atendimento em 7 pedidos.
- **entragaPedido:** a cada pedido pronto, este deve se entregue ao cliente, a considerar a origem. Sendo assim, os de DriveThru e Balcão são entregues imediatamente. Já o delivery, deve-se acumular 3 pedidos para efetuar a entrega.

Sistema de Agendamento (Gustavo, Tales)
Esse sistema simula o controle de agenda médica. Um consultório pode ter 1 ou N médicos, cada médico tem um CRM e uma agenda. Todos os médicos realizam atendimento apenas nos dias úteis (considere meses de 4 semanas) e com uma restrição de 5 consultas por dia ou 4 consultas e 1 cirurgia.
A simulação deve conter as funções:

- **agendar:** Essa função tem por propósito adicionar um compromisso, consulta (1) ou cirurgia (2), na agenda do médico por CRM. Cada agendamento de compromisso médico, é composto por ID (gerado e controlado sequencialmente pelo simulador), dia do agendamento e número de atendimento no dia (1 á 5). No entanto, para agendar um compromisso, a função recebe o CRM do médico, o tipo de compromisso (consulta ou cirurgia) e o dia desejado do compromisso.
- **cancelarAgendamento:** Para cancelar um compromisso é necessário informar o dia e o número do atendimento, ficando disponível para um novo atendimento.
- **otimizarAgenda:** Essa função deve otimizar os compromissos da agenda de um médico. A otimização reagendará os compromissos, antecipando os compromissos conforme a política, preenchendo os horários vagos.

Processo de Matrícula (Leonardo, Hector e Thiago Araujo)
Implemente um algoritmo que simule o contexto de uma secretária escolar. Durante o período de matricula os alunos (RA, nome, série e nota) são organizados em turmas. Com início das aulas o professor solicitar o diário de classe da Turma e ao decorrer lança as notas dos respectivos alunos. Por fim, ao final do curso, a secretaria disponibiliza um relatório de Aprovados e Reprovados.

A simulação deve conter as funções:

- **realizarMatricula:** a matrícula de um aluno será representada por essa função que recebe o Nome e série do aluno. A função deve gerar o RA do aluno (que é definido pelo código da série seguido do número de matriculados na série; Ex. RA=0218, o aluno matriculado está na segunda série [02] sendo decimo oitavo matriculado na turma).
- **criaDiario:** informando a séria, a função deve criar uma lista de alunos matriculados para a séria em ordem alfabética.
- **lancaNota:** com base no RA do aluno, a nota do respectivo aluno deve ser atualizada.
- **aprovadosResprovados:** essa função deve ser um relatório que informa, em ordem alfabética e separadamente, quais os alunos aprovados (nota >= 6.0) e reprovados (nota < 6.0)

Escalonamento de Processos (Classio, Thais e Letícia)
Implemente um algoritmo que simule o escalonamento de processos de um sistema operacional. A simulação é composta por um processo, contendo os atributos ID e Número de Ciclos. A gestão da política do Escalonador deve ocorrer uma fila (limitado em 10 processos). Sendo uma fila, o Escalonador deve operar sobre FIFO. A simulação deve conter as funções:

- **adicionarProcesso:** recebendo apenas o número de ciclos, o ID será definido pelo simulador (sendo este único). O processo deve ser adicionado no final da fila. A função deve informar caso esteja cheia.
- **executarProcesso:** O processo que está na primeira posição da fila deve ser "processado" (simbolizado pelo decremento do número de ciclos do processo). Após a execução, o processo deve ir para o final da fila caso tenha ciclos á executar.
- **estadoEscalonador:** A função deve informar o estado do escalonador, quantos processos tem na fila e o ID que está em execução.

Bingo (Felipe, Luis e Luiz)
Simule, por meio de um algoritmo, um sistema de aposta. Cada apostador (no máximo 30) tem uma cartela com 9 únicos números (entre 1 e 100). Ao iniciar o bingo, um número é sorteado (entre 1 e 100). A cada sorteio deve-se verificar o acerto do número na cartela de cada apostador (desconsidere o ato "comer barriga"). No final de cada rodada, após o sorteio e a verificação, deve-se apresentar uma lista em ordem de maior número de acertos entre os apostadores. O primeiro apostador que completar a cartela vence o Bingo, este deve ser informando como ganhador.
A simulação deve conter as funções:

- **gereCartelas:** dado um número, entre 1 e 30, gere as cartelas dos apostadores com valores aleatórios e não repetidos na cartela. Novos apostadores, considerando menos que 30, podem entrar durante o bingo.
- **rankingDeAcertos:** essa função deve contabilizar e construir o ranking com de acertos de cada apostador.
- **sortearNumero:** essa função deve sortear um número (1 e 100) e verificar o acerto entre as cartelas dos apostadores. Quando um apostador completar a cartela, o simulador deve informar quem é o vencedor.

Tradutor de Código Morse (Bruno, Gabriel e Renner)
Crie um algoritmo que faça as traduções de uma palavra (em português) para código Morse (apenas letras e números). Crie uma struct que tenha os atributos Letra e Código. As traduções devem desconsiderar os caracteres especiais e No Case Sensitive (não fazer distinção de letras maiúscula e minúsculas).
A simulação deve conter as funções:

- **traduzirMorse:** data uma sequência de Códigos Morse, traduza-o para o Português.
- **traduzirPortugues:** data uma sequência de letras (sendo uma palavra ou frase em português), traduza-o para o Código Morse.

- **ordenar:** dado uma séria de código morse crie uma função que seje capaz de ordenar a série em ordem decrescente.