UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
Rua Dom Manoel de Medeiros, s/n – Dois Irmãos 52171-900 Recife-PE
Fone: 81 3302 1000 www.dc.ufrpe.br

DISCIPLINA: Algoritmos e Estruturas de Dados **CÓDIGO**: 06214
DEPARTAMENTO: Computação **ÁREA**: Informática
CURSO: Licenciatura em Computação
PROFESSOR RESPONSÁVEL: Luciano Demétrio Santos Pacífico
DATA MÁXIMA DE ENTREGA: **19-09-2020**

Regras do Projeto 01 – 1ª Verificação de Aprendizagem

1. **Não é permitido o uso de Estruturas de Dados prontas de Linguagens de Programação**. O aluno deve implementar suas próprias Estruturas de Dados. Na Linguagem de Programação **C**, deve-se usar **structs**. Nas demais Linguagens de Programação permitidas (vide **Regras da Disciplina**), deve-se usar **classes**.
2. **Não é permitido o uso de Algoritmos e comandos otimizados prontos de Linguagens de Programação**. Todos os algoritmos solicitados devem ser implementados pelos alunos como **procedimentos** (funções, métodos, etc.).
3. As questões que solicitam **escrita de código** devem ser resolvidas **apenas através dos recursos oferecidos pela pseudolingaugem definida para a disciplina, e dos recursos equivalentes em Linguagens de Programação reais**, sendo eles: variáveis, constantes e tipos primitivos, expressões, estruturas condicionais, estruturas de repetição, sub-rotinas, estruturas de dados homogêneas (Arrays) e estruturas de dados heterogêneas (registros – classes e sctructs).
4. Para o Projeto 01, os **Arrays** devem ser implementados através de **Alocação Estática**. Não será permitido o uso de **Estruturas de Dados dinâmicas** implementadas em Linguagens de Programação, como os **Vectors** e **Lists** da Linguagem de Programação **Java**, por exemplo. Deve-se usar **arrays**.
5. Como a Linguagem de Programação **Python** não suporta **arrays estáticos**, o aluno que optar por usar esta Linguagem deverá usar **Lists**, **única e exclusivamente para simular o comportamento de arrays estáticos**, de acordo com os seguintes critérios:
    a. Deve-se criar uma **List** com posições vazias através do método **append** dessa estrutura. O uso do append será permitido **apenas na alocação de memória para a variável que representará o array estático**;
    b. O limite máximo de M objetos deve ser controlado **através de código**;
    c. A Estrutura deve ser manipulada como se fosse um **array estático**, com os **procedimentos escritos pelo aluno**, **não sendo permitido** o uso de métodos, funções ou otimizações oferecidos pela Linguagem Python.
6. **Regra de Ouro**: **Todos os alunos envolvidos em cópias terão suas notas ANULADAS nas referidas questões.**
7. Apenas o código ".c", ".cpp", ".java", ".py", etc. deve ser enviado ao professor para cada questão. Deve-se enviar **um único arquivo resposta por questão**, que conterá todas as classes/estruturas e procedimentos necessários para a solução da questão. Todos os arquivos devem ser enviados **em uma única pasta, "zipados"**.
8. O arquivo de resposta com o código do projeto deve ser **nomeado** na forma "P01.c", "P01.java", etc.
9. A resposta do Projeto 01 deve ser submetida **unicamente através da tarefa criada no Google Classroom para este propósito**.

Projeto 01 – 1ª Verificação de Aprendizagem

Deseja-se elaborar um **Sistema de Controle** para um **Jogo de Ação Baseada em Turnos**. Tal sistema deverá ser implementado em uma **Linguagem de Programação real**, de acordo com as especificações abaixo.

PRELIMINARES

1. O Sistema de Controle deverá suportar **exatamente dois times de personagens rivais** (lutadores).
2. Cada time poderá possuir **um número diferente de lutadores**, que poderá **aumentar ou diminuir** ao longo do combate (vide **Funcionamento**).
3. **Não há limite para o número de lutadores em cada time**.
4. Cada **lutador** se caracteriza por um **identificador único**, **seu time**, **valor de dano**, **número de pontos de vida**, e **valor base de iniciativa**.
5. O **valor base de iniciativa** de cada lutador deve ser um valor **inteiro** entre 1 (um) e 100 (cem), inclusive.
6. O jogo será realizado em **turnos**.
7. Cada turno de jogo será caracterizado por **três etapas**: **Organização dos Times**, **Combate** e **Resultados** (vide **Funcionamento**).
8. **Um lutador é considerado vivo se seu número de pontos de vida for maior do que 0 (zero), e morto, caso contrário**.
9. **O jogo será finalizado quando não houver mais lutadores vivos em ao menos um dos times**.

FUNCIONAMENTO

O Sistema de Controle do jogo deve oferecer um **menu interativo ao usuário**, que lhe permita executar as ações no decorrer da execução do jogo, de acordo com o especificado em cada fase.

Fase de Organização dos Times

A Organização dos Times é a primeira etapa em cada turno do jogo. **O jogo já será inicializado nesta etapa**. A Organização dos Times deve permitir que o usuário forneça os seguintes tipos de comando:

1. **Inserção de lutadores em times**: um novo lutador pode ser criado e atribuído a qualquer um dos times. O Sistema de Controle deve assegurar que não haverá tentativa de inserção de lutadores com identificadores iguais (não é permitido jogadores com identificadores iguais no mesmo time, em times diferentes, ou na lista de lutadores mortos). Uma mensagem de erro deve ser impressa, caso o usuário tente fazer inserção de um lutador com um identificador já existente.
2. **Relatório de status de um time**: o sistema deve permitir que o jogador possa acessar o **status atual de um dos times**, avaliando quantos lutadores do time estão em vivos ou mortos, assim como a situação dos lutadores. A lista de jogadores vivos deve ser impressa (apenas seus identificadores, iniciativas e número de pontos de vida atuais) em ordem decrescente de iniciativa, e antes da lista de lutadores mortos. A lista de lutadores mortos deve ser impressa em ordem decrescente de iniciativa. Como entrada, o usuário deve fornecer o número do time desejado.
3. **Fuga de lutador**: o usuário pode decidir remover um lutador de combate (independentemente do time no qual o mesmo se encontre). Para isso, o usuário deve fornecer o identificador do lutador a ser removido, e tal lutador deve estar vivo.

Após as ações do usuário (inserção de lutadores, consulta aos times, ou remoção de lutadores), o Sistema de Controle deve realizar a organização dos times propriamente dita. Para isso, cada time será organizado em uma Fila individual (uma por time), na qual os lutadores estarão ordenados em ordem decrescente de iniciativa. Após a disposição dos lutadores na fila de seus times, o Combate terá início.

Fase de Combate

O Combate será realizado de forma automática. O Fase de Combate é executada da seguinte forma:

1. Os primeiros lutadores em cada fila de time combatem entre si, de acordo com o descrito abaixo:
    a. Cada lutador é **removido da fila de seu time**.

b. Se o lutador **não atacou neste turno**, ele **deve atacar**, causando **dano** ao lutador do time rival.
c. Os lutadores se atacam **simultaneamente** (se aptos), ou seja, os danos causados por cada um será calculado **ao mesmo tempo**. O dano que um lutador causa ao outro deve ser **subtraído da reserva de pontos de vida do lutador atacado**.
d. Se um (ou ambos) dos lutadores combatentes alcançar um número de pontos de vida **menor ou igual a 0 (zero)**, ele será considerado **morto**.
e. Após o ataque mútuo, cada jogador deverá ser **reinserido na fila de seu time** (se estiver **vivo**), ou, se **morto**, deve ser inserido no **Cemitério de seu time**.
f. Cada time terá um **Cemitério próprio**, representado por uma **Lista Ordenada em ordem decrescente**, de acordo com a **iniciativa** dos lutadores mortos.

2. Esse processo deve ser repetido até que **todos os lutadores de todos os times tenham atacado uma vez**, ou **até que algum time fique vazio** (sem lutadores vivos).

Fase de Resultados

A Fase de Resultados será executada **automaticamente**. Ao final de cada Fase de Combate, deve-se calcular o **score de cada time**, sendo tal score **igual à quantidade de lutadores no cemitério do time adversário**.

Nesta fase, as **Condições de Término** serão avaliadas:

1. Se um time possuir ao menos um lutador vivo neste turno, enquanto o time adversário não possui lutadores vivos, este time deve ser declarado vencedor, e o jogo encerrará sua execução.
2. Se ambos os times possuírem lutadores vivos neste turno, e apenas um dos times alcançou score maior ou igual a 20 (vinte), esse time deve ser declarado vencedor, e o jogo encerrará sua execução.
3. Se ambos os times possuírem lutadores vivos, e ambos alcançaram score maior do que vinte neste turno, o time com maior score será considerado vencedor, e o jogo encerrará sua execução.
4. Se ambos os times ficarem vazios neste turno, o time com maior score será considerado vencedor, e o jogo encerrará sua execução.
5. Se ambos os times ficaram vazios neste turno, e o score dos times for igual, o jogo encerrará em condição de empate.
6. **Caso nenhuma das Condições de Término acima seja alcançada, um novo Turno de Jogo deve ser iniciado**.

Todas as Estruturas de Dados e Algoritmos devem ser implementados de acordo com o especificado.