Trabalho final (20 pontos)
Data de entrega: 14/02/2025

A entrega deve ser feita pelo SIGAA até as 23:59 do dia 14/02/2025. Trabalhos entregues entre os dias 15 e 16 terão um desconto de 5 pontos na nota final. Dia 17 começam as apresentações, que serão feitas em uma ordem que será em sala de aula.

Resgate Espacial em Órbita

Objetivo:
Desenvolver um programa em **Java** ou **C++** que simule o resgate de astronautas presos em diferentes módulos de uma estação espacial em órbita. O programa deve aplicar conceitos de orientação a objetos, incluindo herança, polimorfismo, encapsulamento e uso de listas dinâmicas para gerenciar informações relevantes. O aluno deve usar uma linguagem **diferente** da usada no primeiro trabalho. Caso seja feito na mesma linguagem, o trabalho será avaliado pela metade.

Descrição do problema:

Uma estação espacial foi danificada por meteoros, e alguns astronautas estão presos em módulos específicos. Um robô de resgate foi ativado para percorrer a estação, resgatar todos os astronautas e levá-los ao módulo de segurança.

Além de realizar o resgate, o robô deve manter uma **lista dinâmica** com os seguintes dados dos astronautas resgatados:

- Nome (string).
- Nível de saúde (inteiro de 0 a 100).
- Necessidade de atendimento médico urgente (valor booleano).

Essa lista será usada para gerar um relatório ao final da execução.

A estação espacial é representada por uma matriz 2D, onde cada célula representa um módulo. Cada módulo pode conter obstáculos, vazios ou astronautas a serem resgatados. O robô deve seguir as seguintes regras:

1. O robô começa no módulo de segurança (indicado por S) e pode mover-se para cima, baixo, esquerda ou direita.
2. Obstáculos (#) bloqueiam o movimento.
3. Vazios (~) não podem ser atravessados.
4. Cada célula pode conter um astronauta (A). O robô deve coletar todos os astronautas antes de retornar ao módulo de segurança.
5. O robô não pode visitar uma célula que esteja a menos de um módulo de distância de uma célula com fogo (F).

Entrada:
O programa deve processar até 10 arquivos de entrada, nomeados como entrada1.txt, entrada2.txt, ..., entrada10.txt. O relatório correspondente a cada arquivo será impresso na tela e salvo em arquivos de saída nomeados como saida1.txt, saida2.txt, ..., saida10.txt.

Um arquivo contendo os casos de teste:
• Primeira linha: dimensões da matriz N (linhas) e M (colunas).
• As próximas N linhas representam a matriz da estação espacial.

Os caracteres possíveis na matriz são:

- S: módulo de segurança (ponto de partida e chegada do robô).
- A: módulo com um astronauta.
- #: módulo inacessível (obstáculo).
- ~: módulo vazio (vácuo espacial).
- F: módulo em chamas (perigo).
- .: módulo seguro e transitável.

- Após a matriz, uma lista contendo os astronautas no formato:
  - Nome,Nível de Saúde,Atendimento Médico Urgente (0 ou 1)

Saída:
Para cada caso de teste, o programa deve imprimir:

1. O número de passos necessários para resgatar todos os astronautas e retornar ao módulo de segurança deve ser calculado. Os estudantes que implementarem um algoritmo capaz de encontrar o menor número de passos para o resgate poderão ganhar até 3 pontos extras, desde que sejam capazes de explicar claramente a lógica e o funcionamento do algoritmo desenvolvido.
2. Caso não seja possível resgatar todos os astronautas, deve ser informado, ao fim da execução do programa, quais astronautas não puderam ser resgatados.
3. Um relatório final com:
   - O número e a lista de astronautas resgatados (nome, nível de saúde, necessidade de atendimento médico).
   - A lista dos astronautas não resgatados
   - O tempo total da operação de resgate.
   - O estado de saúde de cada um dos astronautas

Exemplo de entrada:

10 10

S.....

.###..#..A

.###..#...

.~~~..F..A

.~~~..F.A.

A~~~..F.F.

.~~~..F...

.....F....

..A..F....

.....

Astronautas:

Yuri Gagarin,85,0

Neil Armstrong,60,1

Buzz Aldrin,95,0

Marcos Pontes,50,1

Valentina Tereshkova,70,0

Posições dos astronautas na matriz:

Yuri Gagarin: (1,9)

Neil Armstrong: (4,8)

Buzz Aldrin: (5,0)

Marcos Pontes: (8,2)

Valentina Tereshkova: (3,9)

A tabela abaixo ilustra o mapa da entrada ao lado. Ela é meramente ilustrativa:

S	.	.	.	.	.	.	.	.	.
.	#	#	#	.	.	#	.	.	A
.	#	#	#	.	.	#	.	.	.
.	~	~	~	.	.	F	.	.	A
.	~	~	~	.	.	F	.	A	.
A	~	~	~	.	.	F	.	F	.
.	~	~	~	.	.	F	.	.	.
.	.	.	.	.	F	.	.	.	.
.	.	A	.	.	F	.	.	.	.
.	.	.	.	.	.	.	.	.	.

Exemplo de saída:

Relatório de Resgate:

- Número de astronautas resgatados: 4

- Buzz Aldrin: Saúde 95, Atendimento Médico Urgente: Não, Posição: (5, 0)

- Marcos Pontes: Saúde 50, Atendimento Médico Urgente: Sim, Posição: (8, 2)

- Yuri Gagarin: Saúde 85, Atendimento Médico Urgente: Não, Posição: (1, 9)

- Valentina Tereshkova: Saúde 70, Atendimento Médico Urgente: Não, Posição: (3, 9)

- Lista de astronautas não resgatados:

- Neil Armstrong: Saúde 60, Atendimento Médico Urgente: Sim, Posição: (4, 8)

- Tempo total da operação de resgate: 44 passos

Requisitos a serem observados:

1. **Orientação a Objetos:**
    - Crie classes como EstacaoEspacial, Modulo, RoboDeResgate, Astronauta, etc.
    - Utilize herança para modelar tipos específicos de módulos (ex.: ModuloSeguranca, ModuloComFogo).
    - Aplique polimorfismo para permitir que o robô interaja com diferentes tipos de módulos.
2. **Lista Dinâmica:**
    - Utilize listas para gerenciar os astronautas resgatados.
    - A lista deve permitir adição de novos elementos dinamicamente e ser iterável para gerar o relatório final.
3. **Algoritmo de Busca:**
    - Use algum algoritmo para encontrar o caminho até cada astronauta.
4. **Tratamento de Exceções:**
    - Lide com erros, como arquivo de entrada inválido ou matriz mal formatada, arquivo de entrada inexistente.
5. **Saída:**
    - Atualize o estado da matriz a cada passo para mostrar o progresso do robô.
    - Apresente estatísticas detalhadas a cada passo, incluindo o total de passos realizados e a quantidade de astronautas resgatados até o momento.

Critérios de Avaliação:

- **Implementação correta** do programa.
- **Uso adequado** de conceitos de orientação a objetos.
- **Integração** da lista dinâmica no programa.
- **Eficiência** do algoritmo de busca.
- **Clareza e organização** do código.

- **Apresentação e explicação** do trabalho.
    - **Nota zero** será atribuída se o aluno não souber explicar o funcionamento do programa, mesmo que ele esteja funcionando corretamente.

Abaixo, segue um exemplo de diagrama de classes para facilitar o entendimento. Provavelmente vocês terão que modificá-lo para desenvolver o programa. Caso modifique-o, entregue junto ao trabalho final o diagrama atualizado: