O que é o projeto:
Basicamente é implementação do famoso jogo clássico de celular *Snake.* Nessa versão a cobra se move em quatro direções usando as setas do teclado, a cobra morre ao colidir consigo mesma ou com uma das bordas da tela.

Como rodar o projeto:
$ stack build
$ stack run

Dificuldades, surpresas e destaque do seu código:
Houveram várias surpresas ao longo do projeto, a primeira delas é a movimentação da cobra, que de inicio pareceu mais fácil e funcionou de primeira mas depois mostrou sua verdadeira face. Outra surpresa foi a aleatoriedade em haskell que deixei para ser implementado no final e não sabia que seria tão complicado de implementar.

- **Movimentação:**
  Para entender ela é preciso entender como a cobra funciona. A cobra é composta por blocos ou nós (Node como está definida no projeto), ela possui uma cabeça, uma cauda e uma velocidade. Cada nó possui uma direção (Norte, Sul, Leste e Oeste) e coordenada em x e y. A cobra se movimenta em na direção da cabeça e na velocidade definida, a princípio havia uma dúvida se a movimentação seria melhor se fosse em pixels ou em blocos. No começo foi tentado implementar por pixels, mas isso levou há muitos bugs quando foi implementado a colisão da cobra com ela mesma, essa colisão é definida de maneira bem simples, basta a cabeça da cobra estar em uma distância menor que o tamanho do bloco de um dos nós da cauda, porém se ela se movesse com pixels acontecia vários casos de a cobra se chocar com sua cauda sem motivo aparente (principalmente quando dava uma volta de 180 graus). Por isso a movimentação teve de ser trocada pra ser em blocos, dessa forma o primeiro nó da cauda herda a posição antiga da cabeça da cobra, o segundo nó herda o do primeiro e assim por diante. O ponto negativo disso é que a velocidade da cobra não fica tão configurável, como o tamanho do nó são 10 pixels por frame, o jogo só fica jogável em 30 frames por segundo, à 60 fps a cobra se moveria muito rápido e à trinta quadros ela parece ter um tempo de resposta mais lento. Uma alternativa é diminuir o tamanho do bloco pra 5 pixels e deixar em 60 quadros por segundo mas o tamanho da cobra na tela fica minúsculo sendo difícil de faze-la comer uma semente.

- **Aleatoriedade:**
  Por Haskell ser puramente funcional gerar um número randômico na linguagem acaba quebrando esse paradigma, já que uma função *f* que recebesse a e b retornaria um resultado diferente em várias chamadas passando os mesmos argumentos *a* e *b*. Pra isso em Haskell existe a *monad IO* para isolar a impureza do código, o problema foi que a aleatoriedade foi deixada pro final do projeto, quando tudo estava implementado, ela seria usada para gerar uma nova coordenada para a semente que a cobra come para crescer, isso faria com que mudasse o retorno de várias funções já implementadas, elas deixariam de retornar um tipo *a* para retornar um tipo *IO a*. Foi optado então por isolar a aleatorieadade no *main* assim como o jogo é implementado em um módulo a parte ele apenas recebe um número gerado aleatoriamente dentro do main para gerar coordenadas pseudo-randômicas para a

semente.

Video: https://youtu.be/CoYIfFXYc9U