**Programação ImperaGva e Funcional** 

**Avaliação o3**

**Atenção:**

• **Para as questões em Haskell, o arquivo correspondente de cada questão deverá iniciar com a letra** q **em minúsculo, seguida pelo número da questão, *underline*, as iniciais do e-mail em minúsculas e com extensão** .hs **Exemplo: considerando que o e-mail é est@cesar.school, o arquivo da terceira questão será** q3\_est.hs **. Qualquer arquivo enviado fora desse padrão será automaGcamente descartado;**

• **Para as questões em C, o arquivo correspondente de cada questão deverá iniciar com a letra** q **em minúsculo, seguida pelo número da questão, *underline*, as iniciais do e-mail em minúsculas e com extensão** .c **Exemplo: considerando que o e-mail é est@cesar.school, o arquivo da primeira questão será** q1\_est.c **. Qualquer arquivo enviado fora desse padrão será automaGcamente descartado;**

• **Os arquivos deverão ser salvos dentro de um diretório cujo nome será formado pelas iniciais do e-mail em minúsculas. Exemplo: considerando que o e-mail é est@cesar.school, o diretório será** est **Qualquer diretório enviado fora desse padrão será automaGcamente descartado;**

• **O diretório será compactado em um arquivo**.zip**, e este deve ser disponibilizado para o professor via Classroom. O nome do arquivo será formado pelas iniciais do e-mail em minúsculas. Exemplo: considerando que o e-mail é est@cesar.school, o arquivo a ser entregue será** est.zip . **Qualquer arquivo enviado fora desse padrão será automaGcamente descartado;**

• **Para as questões em Haskell, não serão válidas questões que uGlizarem a importação de módulos;**

• **Para as questões em Haskell, todas as funções devem apresentar o cabeçalho correspondente;**

• **IdenGficada a cópia de qualquer questão, seja com relação a outre alune ou de alguma outra fonte, TODA a avaliação será descartada;**

• **As questões em C que não seguirem o padrão de codificação do kernel do Linux sofrerão um decréscimo em 20% da nota final;**

• **As questões em C que apresentarem vazamento de memória sofrerão um decréscimo em 20% da nota final.**

1. (2.5) Uma palavra “aninhada" é construída a par<r de uma ou mais adições de uma palavra semente "dentro" da mesma. Por exemplo

semente: school

Primeiro aninhamento: schschoolool

Segundo aninhamento: schsschoolchoolool

semente: school

Não é um aninhamento: schoolschool

Assim, dada uma semente e uma palavra, ambas por linha de comando, escreva um programa em C que determine se a palavra é aninhada com relação a semente, retornando True, ou não, retornando False. Não u<lize nenhuma outra biblioteca além da stdio.h .

./q1\_login school schsschoolchoolool

True

./q1\_login school schoolschool

False

Conteúdo relacionado: ponteiro, funções

2. (2.5) Um labirinto pode ser representado por uma matriz 2D, em que 0 (zero) representa um caminho aberto e 1 (um) está associado a uma parede. Iniciando do canto superior esquerdo e com a saída correspondente ao canto inferior direito, construa um programa em C que lê um arquivo com um labirinto formado por 0’s e 1’s e indique se é possível percorrer o labirinto. São permi<dos apenas movimentos para cima, para baixo, para a esquerda e para a direita. O nome do arquivo será passado por linha de comando.

Exemplo 1: considere o arquivo matriz.txt com a seguinte matriz

0 1 1 1 1 1 1

0 0 1 1 0 1 1

1 0 0 0 0 1 1

1 1 1 1 0 0 1

1 1 1 1 1 0 0

A chamda deve ser:

./q2\_login matrix.txt

True

Exemplo 2: considere o arquivo matrix2 com a seguinte matriz

0 1 1 1 1 1 1

0 0 1 1 0 1 1

1 0 0 0 0 1 1

1 1 1 1 0 0 1

1 1 1 1 1 0 1

A chamda deve ser:

./q2\_login matrix2

False

Conteúdo relacionado: leitura de arquivo, matriz

3. (2.5) Implemente permute em Haskell que dada uma lista, retorne todas as permutações possíveis dos seus elementos, sem repe<ção e em ordem crescente. Não u<lize compreensão de lista e pode-se aplicar apenas as funções predefinidas em Haskell que foram apresentadas em sala de aula.

\*Main> permute []

[[]]

\*Main> permute "ab"

["ab","ba"]

\*Main> permute "aba"

["aab","aba","baa"]

Conteúdo relacionado: lista

4. (2.5) . A notação pós-fixada é uma representação matemá<ca em que cada operador de uma expressão se encontra após todos seus operandos, em contraste com a notação pré-fixada, em que os operadores aparecem antes dos operandos. Uma caracterís<ca importante dessa notação é que ela é livre de parênteses enquanto as paridades dos operadores forem fixas. Sua tarefa será escrever uma função onp :: String -> String que transforma expressões algébricas com parênteses em expressões na forma pós-fixada. Todos os operadores envolvidos são binários: +, -, \*, /, ^ (ordem de prioridade do menor para o maior). Os operandos serão exclusivamente letras do alfabeto de a à z. Assuma que haverá apenas uma forma pós-fixada (sem expressões como a\*b\*c). Para esclarecer, alguns exemplos:

\*Main> onp "a+(b\*c)"

"abc\*+"

\*Main> onp "(a+b)\*(z+x)"

"ab+zx+\*"

\*Main>

Não u<lize compreensão de lista e pode-se aplicar apenas as funções predefinidas em Haskell que foram apresentadas em sala de aula.

Conteúdo relacionado: lista