

# Lista de Exercícios 1 – INF05008

- USE OS NOMES DE FUNÇÕES DEFINIDOS NAS QUESTÕES.
- Use o **template da solução disponível no Moodle**.
- Nas questões 1 a 3, DEVE ser colocada a documentação completa, ou seja, contrato, objetivo, exemplos E testes (não use testes como exemplo nesta lista).
- Não precisa colocar testes nas funções definidas nas questões 5 e 6. Nos exemplos, ou cole a imagem gerada ou explique que imagem deve ser gerada em cada exemplo.
- Nesta lista será avaliada também a organização do código: uso de indentação apropriado e de comentários. O código deve ser o mais legível possível!
- Se uma função retorna uma String, fique atento ao modelo de String que foi pedido na questão. A string retornada deverá ser exatamente igual ao solicitado (espaços, acentos, etc).
- Salve seu programa em um arquivo e envie pelo Moodle. O nome do arquivo deve ter o seguinte formato:

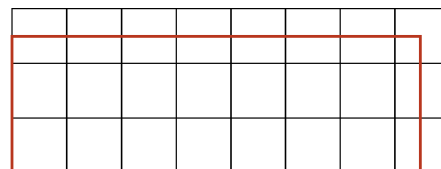
**lista1-*NomeDoAluno-TurmaDoAluno.extensao***

onde em *NomeDoAluno* você deve colocar o seu nome, em *TurmaDoAluno* você deve colocar a sua turma (A, B, C ou D) e a *extensao* deve ser rkt (se você usou o DrRacket) ou txt (se você usou o WeScheme). Não envie arquivos em outros formatos. Não coloque espaços nem acentos no nome do arquivo.

1. Monte uma função chamada **número-azulejos** que, dadas as medidas de uma parede (largura e altura) e o tamanho do lado de um azulejo quadrado (todos em cm), nesta ordem, devolve o número de azulejos necessários para cobrir a parede. Assuma que, se precisar cortar um azulejo, o resto que sobrar poderá ser utilizado para cobrir a parede também e que só é possível comprar um número inteiro de azulejos, portanto o resultado deve ser o menor número inteiro de azulejos que cobre a parede.
2. Depois de falar com o azulejista, descobrimos que seria muito difícil não quebrar alguns azulejos na hora de cortar, o que provavelmente inviabilizaria utilizar de modo ótimo os azulejos, como foi calculado na questão anterior. Monte uma função chamada **número-azulejos-com-sobra** que, dadas as medidas de uma parede (largura e altura) e o tamanho do lado de um azulejo quadrado (todos em cm), nesta ordem, devolve o número de azulejos necessários para cobrir a parede, considerando que se houver necessidade de cortar um azulejo, somente uma das partes poderá ser utilizada. Observe a comparação entre os resultados das funções no exemplo abaixo, bem como a figura, que mostra uma parede em vermelho e o aproveitamento dos azulejos em preto. *Dica: Use a função construída na questão anterior para construir esta nova função!*

(número-azulejos 750 250 100) = 17

(número-azulejos-com-sobra 750 250 100) = 24



3. Imagine agora que, ao invés de retornar um número, gostaríamos de mostrar uma mensagem (string) informando quantos azulejos seriam necessários. Desenvolva o programa **número-azulejos-msg** que, dados os mesmos argumentos da função do exercício anterior, gera a mensagem "O número de azulejos necessários é N.", onde N é o número de azulejos que seriam necessários para cobrir a parede. *Não faça novamente uma função para calcular o número de azulejos necessário, você já fez isto na questão anterior! Apenas use a função que você já fez. Será descontada nota de quem não utilizar a função número-azulejos-com-sobra.*
4. Construa o programa **monta-palavra** que, dadas duas palavras quaisquer P1 e P2, gera a seguinte mensagem "P1-P2 => N letras", onde N é soma dos números de letras de P1 e P2, conforme os exemplos abaixo:

(monta-palavra "A" "B") = "A-B => 2 letras"

(monta-palavra "guarda" "chuva") = "guarda-chuva => 11 letras"

5. Defina uma **constante** chamada **Nave-espacial** que gera a imagem abaixo. Para definir esta imagem, você deve usar pelo menos **uma função e 3 outras constantes** com sub-imagens. O tamanho de cada barra deve ser 100 por 25, e as cores usadas devem ser red, orange, yellow, green, blue e violet. *Dica: Observe os padrões necessários e defina constantes/funções com nomes significativos para minimizar a repetição de código e tornar suas definições mais fáceis de entender.*



6. Defina uma **constante** chamada **Minha-imagem** que gera uma imagem de sua escolha. Para construir esta imagem você deve usar pelo menos 5 outras constantes (definido sub-imagens de sua imagem) e uma função (mas pode usar mais, se quiser). **Indique claramente quais nomes são de constantes e quais são de funções.**
7. Desafio (não vale nota): E se a gente quisesse fazer uma pirâmide colorida como a imagem a seguir, como poderíamos modificar a definição de **Nave-espacial**? Pesquise no manual do pacote de imagens do Racket uma função que pode ser útil para isso: <https://docs.racket-lang.org/teachpack/2htdpimage.html>



Algumas funções pré-definidas úteis:

**ceiling** : Número → Número

Obj: **ceiling** é uma função que recebe um número e devolve o maior número inteiro que contém este número.

Exemplo: (**ceiling** 20.1) = 21

**number->string** : Number → String

Obj: **number->string** é uma função que recebe um número e transforma esse número em uma palavra (string).

Exemplo: (**number->string** 20) = "20"

**string-append** : String ... String → String

Obj: Dados dois (ou mais) strings, junta (concatena) esses strings.

Exemplo: (**string-append** "ABC D " "EF") = "ABC D EF"


**string-length** : String → Número

Obj: Dada uma string, devolve o número de dígitos da string.

Exemplo: (**string-length** "ABC D \$") = 7

**rectangle** : Number Number String String → Image

Obj: Dados os tamanhos dos lados, o tipo de preenchimento e a cor, desenha a imagem do retângulo correspondente.

Exemplo: (**rectangle** 40 20 "outline" "skyBlue") = 

**above** : Image ... Image → Image

Obj: Dadas várias imagens, gera uma imagem com as imagens uma acima da outra, a primeira ficará no topo.

Exemplo: (**above**   ) = 

**beside** : Image ... Image → Image

Obj: Dadas várias imagens, gera uma imagem com as imagens uma ao lado da outra, a primeira ficará no esquerda.

Exemplo: (**beside**  ) = 