Problemas da semana 4

Funções parte 1

Nas questões desta semana vocês devem utilizar funções. Implementar uma função pode dar um trabalho inicial, mas uma vez pronta e testada, todo o resto fica mais fácil. Depois, sempre que precisar, a função estará lá pronta para ser reutilizada para outros propósitos. Mais adiante no curso, na unidade 3, veremos como construir nossas próprias bibliotecas de funções, assim como faz a math.h!

Problema 1 - Horários das rondas

Um grupo de policiais precisa realizar periodicamente algumas rondas para minimizar os impactos da criminalidade no bairro. Horários periódicos como a cada 30 minutos são fáceis de calcular, mas muito previsíveis. Os policiais pediram sua ajuda para listar os horários das rondas em dois formatos possíveis:

1. No formato 24h: hh:mm, com as horas entre 0 e 23 e os minutos entre 0 e 59. Por exemplo:
   a. 00:15
   b. 09:07
   c. 12:53
   d. 22:44
2. Ou no formato 12h: hh:mm AA, com as horas entre 1 e 12 e os minutos entre 0 e 59, onde AA pode ser AM para horários antes do meio-dia e PM para horários depois do meia-dia (consulte https://pt.wikipedia.org/wiki/Sistema_hor%C3%A1rio_de_12_horas). Por exemplo (os mesmos horários do exemplo acima):
   a. 12:15 AM
   b. 09:07 AM
   c. 12:53 PM
   d. 10:44 PM

A entrada do programa consiste em 2 inteiros contendo horas e minutos da primeira ronda (sempre no formato 24h), seguidos de um inteiro, tal que 0 significa escrever os horários das rondas no formato 24h e 1 no formato 12h. O programa deve escrever na saída os horários de todas as rondas obedecendo os seguintes **acréscimos** em relação ao horário da primeira ronda: 1h, 2h10m, 4h40m e 12h5m.

Para resolver essa questão, seu programa deve implementar uma função que escreve um horário na tela, tratando os casos em que as horas e/ou os minutos estão fora do intervalo. Por exemplo, se o horário passado for 25 horas e 10 minutos, a função deve escrever 1 hora e 10 minutos (passou de um dia para o outro) – isso deve facilitar bastante a programação do resto.

Exemplos

Input	Output	Input	Output
10 15 0	10:15	22 55 1	10:55 PM
	11:15		11:55 PM
	12:25		01:05 AM
	14:55		03:35 AM
	22:20		11:00 AM

Figure 1: Policiais esperando você terminar o programa

Problema 2 - Primos triplos

O filho de um amigo seu tem só 12 anos e está encantado com os números primos. Ele gosta tanto que está montando uma lista com todos os trios de primos na forma $(x, x+2, x+6)$. Ele conseguiu encontrar o trio 641, 643, 647, mas levou quase uma tarde toda para encontrá-lo. Ele soube que você está aprendendo programação e é capaz de escrever um programa que escreva todos os trios de primos até 50000 quase instaneamente! É sua hora então de automatizar essa árdua tarefa usando a linguagem C.

A saída do programa deve ser todos os trios de primos na forma $(x, x+2, x+6)$ até 50000, um por linha, no formato (x, x+2, x+6). Ou seja, as 4 primeiras linhas da saída devem ser:

(5, 7, 11)

(11, 13, 17)

(17, 19, 23)

(41, 43, 47)

Para facilitar a programação, você deve necessariamente escrever uma função que retorna se um determinado número passado como parâmetro é primo ou não!

Problema 3 - Pousando a sonda espacial

Você faz parte de uma equipe do INPE que precisa encontrar um local seguro para pousar uma sonda no terreno de uma lua de Saturno. Após algumas análises, seus colegas conseguiram modelar a altura da superfície como uma função bidimensional

$$f(x, y) = sin(cos(y) + x) + cos(y + sin(x))$$

Todas as alturas abaixo de zero e acima de 2 provocam danos para a sonda e devem ser evitadas. Um trecho dessa superfície está na imagem abaixo. A sonda (cujo centro é o ponto branco na figura) possui um certo tamanho e, portanto, todos os 4 pontos (em verde) também devem estar em uma altura permitida. Se a sonda estiver em $p = (x, y)$, esses pontos estarão em $(x + 0.2, y + 0.2)$, $(x - 0.2, y - 0.2)$, $(x + 0.2, y - 0.2)$ e $(x - 0.2, y + 0.2)$. Se todos os 5 pontos estiverem na altura segura, o ponto de pouso é dito seguro.

Como a lua de Saturno possui fortes rajadas de vento, pode ser que a sonda não pouse exatamente no local $p = (x, y)$. Para ter uma maior segurança, você deve verificar também as condições de pouso em $(x + 2, y)$, $(x - 2, y)$, $(x, y - 2)$ e $(x, y + 2)$ (vide os pontos em azul **a, b, c e d** na figura).

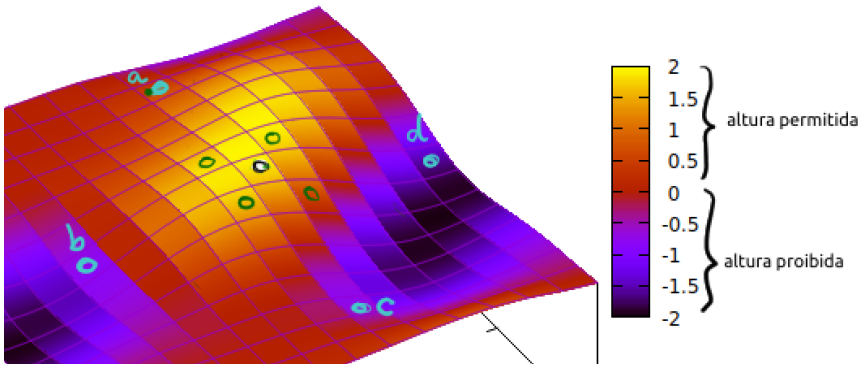

Figure 2: Sonda posicionada no ponto P. Observação: os pontos em verde estão mais distantes na figura do que deveriam para melhor visualização.

Escreva um programa que leia a coordenada de pouso da sonda (x, y) e escreva a classificação do local, de acordo com as seguintes regras:

1. Se esse ponto não for seguro, a classificação é "troque de coordenada"
2. Se esse ponto for seguro, conte quantos dos 4 locais vizinhos são seguros:
   b. 0 ou 1 ponto seguro: "inseguro"
   c. 2 ou 3 pontos seguros: "relativamente seguro"
   d. 4 pontos seguros: "seguro"

Todas as coordenadas devem ser declaradas como float e seu programa deve implementar pelo menos duas funções: uma para calcular a altura do terreno em função da coordenada (x, y) e outra função para verificar se a coordenada (x, y) é um ponto de pouso seguro.

Exemplos

Input	Output
0 0	relativamente seguro
3.1 4.2	troque de coordenada
8.4 4.8	seguro