Primeira Avaliação

Programação I (INF15927) - UFES
18 de Outubro de 2022

Instruções

- A prova é **individual**. Todas as questões serão testadas e plágio não será tolerado
- Guarde o seu celular na sua mochila ou algo similar. O uso de celular durante a prova não será tolerado
- Não custa relembrar, mas todas as questões devem ser desenvolvidas na **linguagem de programação C**
- Organização/indentação e boas práticas de programação é critério de avaliação. Use nome de variáveis de forma inteligente
- **Escolhas de tipo de dados e inclusão de bibliotecas é por sua conta**
- Cada questão possui 10 casos de testes, na qual 4 são públicos e estão disponíveis para vocês. Siga a risca as instruções de cada uma das questões para que o teste automático não falhe
- Seu código será compilado usando o comando `gcc <nome_arquivo.c> -o <nome_arquivo> -lm`
- Seu código será executado em um sistema operacional Linux com o comando `./<nome_arquivo> < <entrada_n> > <saida_n>`
- Você deve criar uma pasta com seu <nome_sobrenome> e colocar todas suas questões dentro dela. O nome dos arquivos das questões são descritos em cada questão.
- **Todas as questões tem o mesmo peso**

Questão 1: Lembretes de senha

Atualmente precisamos memorizar diversos tipos de senhas numéricas, inclusive de bancos. Muitas pessoas tem problemas em memorizar diversos números e acaba anotando a senha em um caderno ou em um bloco de notas. Todavia, isso é bastante inseguro, uma vez que se alguma pessoa encontrar esses números, ela pode obter acesso a sua conta digital. Uma maneira um pouco mais segura é criar um lembrete de senha. Porém, nada impede que, mesmo com o lembrete, a senha seja esquecida. Sendo assim, para auxiliar neste problema, você deve criar um simples programa que seja capaz de armazenar senhas de até 5 números através de um lembrete.

Funcionamento do programa: seu programa deve receber como entrada dois números inteiros, o primeiro representa o `lembrete` de senha, e o segundo a quantidade de `dígitos` da senha. A senha será obtida de acordo com os seguintes passos:

1. Encontre o maior número possível de se formar com os dígitos informados no `lembrete`. Vamos chamar esse número de `maior`

2. Agora faça a subtração `maior` - `lembrete`
3. Agora, para obter a senha, selecionamos os n primeiros dígitos determinados pela quantidade de `dígitos` informado pelo usuário

Exemplo de funcionamento: imagine um `lembrete` de senha igual a 25697 e a quantidade de `dígitos` igual a 4. O `maior` número possível a se formar com os dígitos de `lembrete` é 97652. Fazendo a subtração 97652 - 25697, obtemos 71955. Como a quantidade de `dígitos` é 4, a senha final é **7195**.

Limites:

- $10000 \leq$ `lembrete` ≤ 99999
- $2 \leq$ `dígitos` ≤ 5

Padrão de entrada e saída:

- Entrada: `<lembrete> <dígito>`
- Saída: `<senha>`

Nome do arquivo da questão: `lembrete.c`

Questão 2: Conversão decimal-binária

O sistema de numeração binária é importantíssimo para computação uma vez que computadores só entendem 1 ou 0. Por conta disso, programas que convertem sistemas de numeração (decimal, binário, hexadecimal, etc) são essenciais. Nós, seres humanos, utilizamos o sistema decimal de maneira bem natural (lembre-se, temos 10 dedos). Esse sistema possui 10 algarismos ($\{0, \cdots, 9\}$) que são multiplicados por potências de 10. Por exemplo, imagine o número 5240_{10}, na base decimal, ele é obtido da seguinte forma:

$$5240_{10} = 5 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 0 \times 10^0$$

Um número binário, é obtido de forma similar. Porém, neste sistema temos apenas dois algarismos ($\{0, 1\}$). Para converter um número binário para decimal, usamos uma lógica similar, porém, utilizando potências de 2. Por exemplo:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

Neste caso, o número 1101_2 no sistema binário é igual ao número 13_{10} no sistema decimal. O número 10 ou 2 subscrito ao número principal representa o sistema decimal (também chamado de base 10 ou base 2, para decimal e binário respectivamente).

Cada número no sistema binário, ou seja, 1 ou 0, é chamado de bit. No exemplo anterior, 4 bits são necessários para representar o número decimal 13. Logo, o tamanho máximo em decimal que um número binário consegue representar depende do número de bits. Um número binário de 8 bits consegue representar no máximo $2^8 - 1$, ou seja, um número decimal de 0 a 255. O motivo é simples, o maior número binário com 8 bits é 11111111_2, que em decimal fica:

$$11111111_2 = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 255_{10}$$

Já o menor é 00000000_2, que é fácil visualizar que também é 0 na base 10.

Dado toda essa introdução, sua missão aqui é construir um programa que receba um `número` inteiro decimal e converta-o para um número binário de 8 bits.

Exemplo de funcionamento: supondo que o seu programa receba um número decimal igual a 45, ele deve retornar o número binário 00101101, que é a conversão decimal dele, pois:

$$00101101_2 = 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 255_{10}$$

Limites:

- $0 \leq$ `número` ≤ 255

Padrão de entrada e saída:

- Entrada: `<número_decimal>`
- Saída: `<número_binário>`

Nome do arquivo da questão: `binario.c`

Questão 3: Aproximando uma integral

No cálculo, a integral de uma função foi criada originalmente para determinar a área sob a curva de uma função f no plano cartesiano, como exemplificado na Figura 1.

Para calcular o valor de uma integral numericamente, é necessário utilizar aproximações, uma vez que calcular a área exata como da Figura 1 é impraticável. Uma das regras mais famosas é a regra do trapézio. Ela é bem simples e consiste representar a área total de uma curva através do somatório da área de vários trapézios menores que é simples de calcular. Essa técnica é ilustrada na Figura 2.

Perceba que todo intervalo $[a, b]$ é divido em n trapézios que, ao somar a área de todos eles, temos algo próximo do que seria o valor real da integral. Obviamente teremos um erro de aproximação; porém, esse erro diminui com o aumento do número de trapézios. Isso é conhecido como discretização do intervalo $[a, b]$. Em outras palavras, a distância $x_1 - a$ depende do número de trapézios.

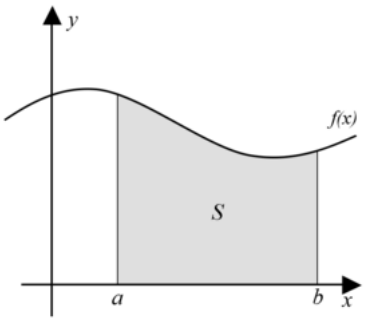

Figura 1: Exemplo da área sob a curva (S) que representa o valor da integral de $f(x)$ no intervalo $[a, b]$

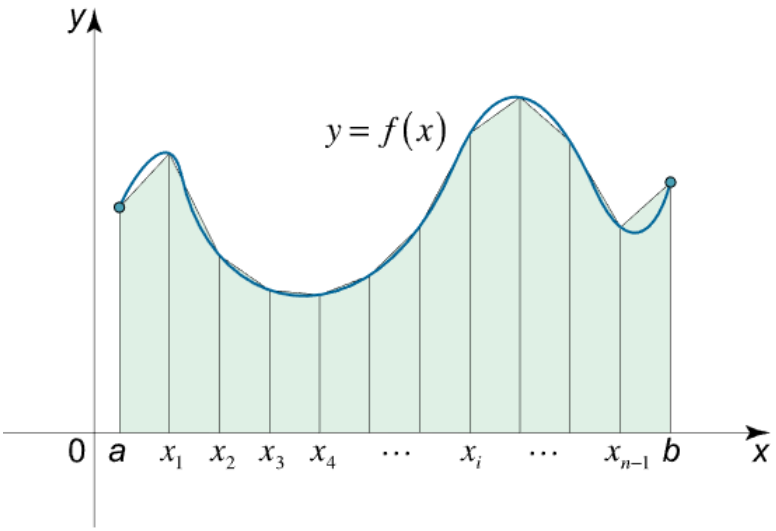

Figura 2: Aproximando o valor de uma integral de $f(x)$ no intervalo $[a, b]$ utilizando a regra do Trapézio. Perceba que a soma das área de todos os trapézios é próximo a integral real

A área de um trapézio é ilustrada na Figura 3. Sabendo disso, faça um programa que receba o intervalo fechado $[a, b]$ e o número de trapézios para calcular a integral aproximada da função $f(x) = sin(x + 1) + \sqrt{x + x^3}$.

Exemplo de funcionamento: imagine que seu programa receba $a = 1$, $b = 6$ e $n = 10$. Você deve calcular o valor aproximado da integral de $f(x)$ utilizando a técnica descrita acima dividindo a curva em 10 trapézios.

Limites:

- $10 \leq$ n ≤ 1000
- $1 \leq$ a,b ≤ 1000
- Utilize precisão simples nos cálculo da integral

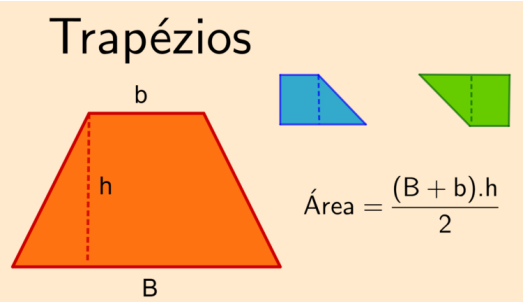

Figura 3: Ilustração da área de um trapézio

- A resposta deve ter 2 casas decimais

Padrão de entrada e saída:

- Entrada: `<a> <b> <n>`
- Saída: `<integral>`

Nome do arquivo da questão: `integral.c`

Questão 4: Mini batalha naval

Batalha naval é um jogo de tabuleiro de dois jogadores, na qual os jogadores têm de adivinhar em quais quadrados estão os navios do oponente. Ganha o jogo quem afundar mais navios do adversário. O tabuleiro é um grid $M \times N$ onde os navios são posicionados. Cada navio pode ser encontrado por uma coordenada indicando a posição da linha e coluna na qual o navio se encontra. Um exemplo de um tabuleiro de batalha naval é ilustrado na Figura 4.

Neste tabuleiro, de 6 linhas e 4 colunas, temos as coordenadas nas laterais (linhas e colunas) e os navios são representados por números (em vermelho). Cada número indica a importância do navio. Um jogador pode atirar um míssil indicando uma coordenada no tabuleiro. Por exemplo, a coordenada (2,3) atingiria um navio de importância 3, o que daria 3 pontos para o jogador.

Sua missão aqui é fazer um programa que simule o jogo de batalha naval. Seu programa deve receber como entrada o tamanho do tabuleiro (`M`,`N`), as coordenadas de mísseis de dois jogadores ((`x1`, `y1`), (`x2`, `y2`)), a posição das peças no `tabuleiro`, que é representado por uma matriz $M \times N$ na qual o valor zero indica não haver navio na posição. Seu programa deve retornar como saída o jogador que venceu a rodada (`J1` ou `J2`) e sua respectiva pontuação. Se houver empate, retorne apenas `EMPATE`. Se a coordenada do míssil de algum for para fora do tabuleiro, o jogador não recebe nenhuma pontuação.

Limites:

- $4 \leq$ M,N ≤ 20

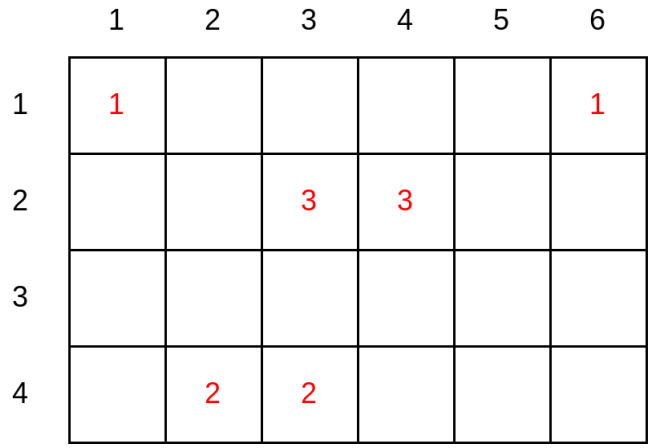

Figura 4: Exemplo de um tabuleiro de batalha naval. Os navios são representados por números (em vermelho) que indicam sua importância.

- As coordenadas dos jogadores são números inteiros
- A importância dos navios nos tabuleiros são representados por números inteiros positivos, sendo zero a ausência de navio

Padrão de entrada e saída:

- Entrada:

```
M N
x1 y1
x2 y2
0 0 ... 0
1 0 ... 3
...
0 0 ... 0
```

- Saída:

```
J{1,2} <pontuação>
```

ou

```
EMPATE
```

Nome do arquivo da questão: `batalha.c`