Lista 1 – Teoria da Computação / Linguagens Formais e Autômatos – 2024/2

Mateus Neves Merçon

29/10/2024

Introdução à Teoria da Computação & Fundamentos Matemáticos

1 Problema 1 — Conversão de Bases Numéricas (3 pts)

Escreva um programa que converte um número inteiro de uma base numérica para outra. O programa deve receber como entrada o número a ser convertido, a base original e a base para a qual ele deve ser convertido. As bases suportadas devem ser 2 (binária), 8 (octal), 10 (decimal), 12 (duodecimal), 16 (hexadecimal) e 20 (vigesimal).

Solução 1

Foi utilizada a função `convertNumber` para realizar a conversão de um número entre bases. A função utiliza dois métodos auxiliares: `toInt` e `fromInt`. A função `toInt` realiza a conversão do número na base de origem para uma representação numérica intermediária, enquanto `fromInt` converte essa representação para a base de destino. A lógica verifica se as bases são suportadas e lança uma exceção caso contrário. O resultado é a string que representa o número convertido na base desejada.

Código-fonte: /problemas/probl1/src/main/scala/Main.scala

2 Problema 2 — Contador de Parênteses Bem Formados (3 pts)

Desenvolva um programa que verifica se uma expressão composta apenas de parênteses (‘(’ e ‘)’) está corretamente balanceada, ou seja, se cada parêntese de abertura tem um correspondente parêntese de fechamento. O programa receberá como entrada uma string contendo apenas parênteses, e como saída uma mensagem que indica se a expressão está bem formada.

Solução 2

Foi implementada a função `checkParentheses` para verificar se uma expressão de parênteses está balanceada. Utilizou-se uma função auxiliar, `balance`, que percorre a expressão recursivamente, incrementando um contador para cada parêntese de abertura e decrementando para cada parêntese de fechamento. Se ao final o contador for zero, a expressão é considerada bem formada. A função lança uma exceção caso sejam encontrados caracteres inválidos. O resultado é uma mensagem indicando se a expressão está ou não balanceada.

Código-fonte: /problemas/probl2/src/main/scala/Main.scala

3 Problema 3 — Números Perfeitos (3 pts)

Escreva um programa que verifique se um número é perfeito. Um número perfeito é aquele que é igual à soma de seus divisores próprios, i.e., os divisores excluindo o um o próprio número. O programa receberá como entrada um número inteiro positivo, e como saída uma mensagem que indica se o número é perfeito ou não.

Solução 3

Foi implementada a função `isPerfectNumber` para verificar se um número é perfeito. A função calcula a soma dos divisores próprios do número, filtrando os divisores válidos e somando-os. Em seguida, verifica-se se a soma dos divisores é igual ao número em questão. Caso seja, o número é considerado perfeito. O resultado é uma mensagem indicando se o número fornecido é ou não perfeito.

Código-fonte: /problemas/probl3/src/main/scala/Main.scala

4 Problema 4 — Cálculo de Expressões com Álgebra Relacional (6 pts)

Crie um programa que calcula o valor de uma expressão da álgebra de conjuntos. O programa deve permitir ao usuário inserir conjuntos (apenas conjuntos definidos por extensão, i.e., listando todos os elementos) e, em seguida, o programa deve receber como entrada uma string que representa a expressão e retornar o resultado. A expressão em questão poderá usar:

- União (`A | B`);
- Interseção (`A & B`);
- Diferença (`A - B`);
- Diferença simétrica (`A ^ B`);
- Complemento (`~A`);
- Produto cartesiano (`A * B`);
- Conjunto das partes (`P(A)`).

Solução 4

Foi utilizada a função `evaluateSetExpression` para avaliar expressões da álgebra de conjuntos. Inicialmente, realizou-se a tokenização e parsing da expressão, convertendo-a em tokens que são processados conforme a precedência das operações. Para cada operador, como união, interseção, diferença e complemento, aplicou-se a operação correspondente sobre os conjuntos envolvidos. O conjunto universo foi definido como a união de todos os conjuntos fornecidos, e o resultado final é o conjunto resultante da expressão.

Código-fonte: /problemas/probl4/src/main/scala/Main.scala