PFL_1

Grupo G03_01:

- Inês Gaspar up202007210
- Pedro Ferreira up202004986

Dependências

> Para correr o nosso programa, é necessário executar o seguinte commando no terminal:

```
cabal install --lib split
```

Inputs para as nossas funções

> Os polinómios têm que seguir o seguinte formato em string: `1*x^3 + 4*x^1*y^3 - 5*z^3 + 6`

> Qualquer outro input que não siga o formato indicado, dará origem a exception. No ficheiro results.hs fornecemos vários polinómios de teste, assim como os seus resultados. Para além disso, no final do mesmo estão as nossas funções que permitem as operações sobre os polinómios.

```
normalizacao polinomio1
adicao polinomio1 polinomio2
multiplicacao polinomio1 polinomio2
derivacao string1 polinomio1
```

Nota: Na derivação apenas se pode fazer a operação em ordem a uma variável singular (por exemplo: x ou y).

Representação Interna

```
type Coef = Integer
type Incognit = [(String, Integer)]

type Monom = (Coef, Incognit)
```

> Escolhemos esta representação para ser mais fácil acedermos à incógnita e ao expoente para as funcionalidades que implementamos.

Funcionalidades

Normalização

> Antes e depois de cada uma das operações que efetuamos, realizamos sempre a normalização do(s) polinómio(s).

Ordenação do monómio por ordem alfabética

> Temos 2 funções que lidam com a parte de ordenação do monómio: primeiro começamos por ordenar apenas as incógnitas (através do algoritmo de ordenação quicksort), depois aplicamos esta função ao monómio e em seguida ao próprio polinómio.

Ordenação do polimónio por grau

> Temos 3 funções que lidam com a parte de ordenação do polinómio: primeiro começamos por ordenar as incógnitas por ordem decrescente do expoente, depois aplicamos a função ao monómio e em seguida (como temos a incógnita de maior grau no primeiro elemento da lista de incógnitas e expoentes) pegamos na primeira incógnita de cada monómio e aplicamos o mesmo algoritmo de ordenação que utilizamos para as incógnitas (quicksort).

Eliminar zeros do polinómio

> Esta função elimina do polinómio, os monómios que têm coeficiente nulo. Para implementarmos esta funcionalidade, bastou-nos aplicar a high order function `filter` ao polinómio onde temos como condição o primeiro elemento do monómio ser nulo (uma vez que na nossa representação o primeiro elemento do monómio corresponde ao coeficiente).

Eliminar incógnitas ^0

> Esta função "apaga" dos monómios as incógnitas que estiverem elevadas a 0. Para tal, a nossa implementação foi a seguinte: para cada incógnita do monómio verificamos se o segundo elemento (o expoente, segundo a nossa representação) é 0, se for substituímos o termo dependente do monómio por o valor default (`[("", 0)]`) para não dar conflitos nas operações que podemos efetuar sobre o polinómio, se não for mantém o valor que tinha.

Adição

> Esta funcionalidade primeiro verifica se os monómios dos dois polinómios têm incógnitas e os expoentes iguais. Se este caso se verificar somamos os termos, se não passamos ao seguinte e fazemos a verificação novamente.

Multiplicação

> Esta funcionalidade faz a combinação de todos os monómios de um polinómio com o polinómio por que o queremos multiplicar, dando-os como resultado. Temos também uma função que verifica se as incógnitas dos dois monómios que estamos a multiplicar são iguais. Se forem mantemos a incógnita e somamos os expoente, se não adicionamos a incógnita e o expoente novo à lista de incógnitas do monómio resultante. Depois aplicamos a função de soma, uma vez que podem resultar monómios com termos dependentes iguais e portanto o resultado pode ser simplificado.

Derivação

> Esta funcionalidade aceita uma incógnita como argumento e o polinómio que queremos derivar. Primeiro verificamos se a incógnita existe em cada um dos monómios do polinómio, caso isso aconteça, multiplicamos o expoente ao coeficiente e subtraímos 1 ao valor do expoente do monómio. Caso não aconteça, mantemos o termo igual. Aplicamos esta função para cada um dos termos do polinómio em questão, através de uma list comprehensions.

Parsing

> Para fazer o parsing do polinómio que recebemos em string para a nossa representação interna implementamos várias funções de separação de carateres. Começamos por eliminar os espaços que podem existir entre os termos do polinómio. De seguida, extraímos o coeficiente e convertemos para Integer (tendo em conta os números negativos) e criamos o tuplo que será cada um dos monómios. Depois, extraímos a incógnita e criamos dentro de cada tuplo a lista que vai conter os tuplos que correspondem ao termo dependente (incógnita e expoente), bem como o expoente. Nesta etapa, tivemos que ter em consideração os casos das constantes e das incógnitas que aparecem sem expoente (que corresponde a ^1). Finalmente, ficamos com a representação interna do polinómio que indicamos anteriormente: `Monom = (Int, [(String, Int)])`.

Reverse Parsing

> Para fazer reverse do polinómio para string fizemos o processo inverso às funções de parsing onde utilizamos várias funções auxiliares. Começamos por juntar as incógnitas com o respetivo expoente, uma vez que pertencem ao mesmo tuple. De seguida, juntamos o sinal ao coeficiente (uma vez que temos que os converter para string). Por fim, basta aplicarmos estas funções ao polinómio na representação interna de onde resulta a nossa string de resultado.