プログラミング言語レポート

氏名: 赤松 佑哉 (akamatsu, YUYA)
学生番号: 09B23595

出題日: 2024 年 5 月 9 日
提出日: 2024 年 5 月日
締切日: 2024 年 6 月 4 日

1 概要

講義を通して学んだ関数型言語 SML 言語を実際の問題解決を通して実践した．今回，C 言語の標準ライブラリに存在する文字列操作関数，strcat , strcmp，strcpy，strexi，strlen , sort と同等の操作をリストに行う SML プログラムを作成した．いかに各関数の概要を示す．

- **listcat** ：strcat に相当し，2 つのリストを連結する．
- **listcmp** ：strcmp に相当し，2 つのリストが要素・順序ともに等しいか判定する．
- **listcpy** ：strcpy に相当し，リストの複製（コピー）を行う．
- **listexi** ：strexi のように，条件を満たす要素がリスト中に存在するかを確認する（高階関数を用いる）．
- **listlen** ：strlen に相当し，リストの長さ（要素数）を返す．
- **listsort**：sort に対応し，マージソートによりリストを昇順に並べ替える．

作成したプログラムは第 9 章に添付している．

2 listcat.sml の実装

二つのリストを引数として受け取り、それらを結合した一つのリストを返す listcat 関数を実装した。パターンマッチングを用いて引数の構造に応じた場合分けを行い、異なる処理を実現している。

片方が空リストであれば、もう片方のリストをそのまま返す。両方のリストに要素がある場合は、第一引数の先頭要素を取り出し、それを再帰的に呼び出された listcat の戻り値の先頭に加えることでリストを構築する。

この再帰処理は、第一引数が空リストとなった時点で終了し、第二引数がそのまま返される。このとき、再帰は打ち切られ、結合されたリストが得られる。

以下のテスト入力により動作確認を行った。

```
val test1 = listcat([1,2,3], [4,5])
val test2 = listcat([1,2,3], [])
```

```
val test3 = listcat([], [1,2,3])
val test4 = listcat([], [])
```

実行結果は次のようになった.

```
$ sml listcat.sml
Standard ML of New Jersey v110.79 [built: Sat Oct 26 12:27:04 2019]
[opening listcat.sml]
val listcat = fn : int list * int list -> int list
val test1 = [1,2,3,4,5] : int list
val test2 = [1,2,3] : int list
val test3 = [1,2,3] : int list
val test4 = [] : int list
```

結果を見るときちんと二つのリストを合併し一つのリストの作成に成功している．また空リストの時もきちんと動作してることがわかる．以上が二つのリストの合併 listcat の実装，解説である．作成した SML ソースコードは第 9.1 章に添付している.

3 listcmp.sml の実装

二つのリストを引数として受け取り，対応する要素の差をすべて合計し返す listcmp 関数を実装した．strcmp 関数のような等価判定ではなく，数値的な差異を合計してリスト間の「違いの度合い」を測ることができる.

関数内部ではパターンマッチングを用い，両方が空リストなら 0 を返す．一方が空リストで他方に要素が残っている場合には，残りの要素をすべて加算して返す．両方に要素がある場合には，先頭要素の差を取り，再帰的に残りのリストに対して同様の処理を行っている.

以下の入力を引数として渡し，動作確認を行った.

```
val test1 = listcmp([1,2,3], [1,2,3])
val test2 = listcmp([1,2,3], [1,2,2])
val test3 = listcmp([1,2,3], [1,2,4])
val test4 = listcmp([1,2,3], [1,2])
val test5 = listcmp([1,2], [1,2,3])
val test6 = listcmp([], [])
```

実行結果は次のようになった.

```
$ sml listcmp.sml
Standard ML of New Jersey v110.79 [built: Sat Oct 26 12:27:04 2019]
[opening listcmp.sml]
val listcmp = fn : int list * int list -> int
val test1 = 0 : int
val test2 = 1 : int
val test3 = ~1 : int
```

```
val test4 = 3 : int
val test5 = 3 : int
val test6 = 0 : int
```

結果を見るときちんと二つのリストの辞書的な差異を数値として出力できている．リストどおしの要素数が違う場合もきちんと動作している．以上がリストを比較する listcmp 関数の実装，解説である．作成した SML ソースコードは第 9.2 章に添付している．

4 listcpy.sml の実装

リストの内容をそのままコピーした新しいリストを返す関数 listcpy を実装した．この関数は，C 言語における strcpy のように元のリストを変更せずに，その要素を順にコピーして新しいリストを構築する機能を持つ．

SML ではリストはイミュータブル（不変）であるため，コピー処理は新しいリストを再帰的に構築する形で実現される．具体的には，パターンマッチングにより，空リストに到達した場合には空リストを返す終了条件とし，非空の場合には先頭要素を新しいリストの先頭に追加しながら再帰呼び出しを行っている．

以下のテストコードにより動作確認を行った．

```
val test1 : int list = listcpy[]
val test2 = listcpy[1,2,3,4]
```

実行結果は次のようになった．

```
$ sml listcpy.sml
Standard ML of New Jersey v110.79 [built: Sat Oct 26 12:27:04 2019]
[opening listcpy.sml]
val listcpy = fn : 'a list -> 'a list
val test1 = [] : int list
val test2 = [1,2,3,4] : int list
```

この結果より，空リストに対しても，任意の整数リストに対しても正しくコピーされたリストを返すことが確認できた．このように，再帰を用いた基本的なリスト操作を通じて，SML の関数型プログラミングの特性を活かした実装を行った．作成した SML ソースコードは第 9.3 章に添付している．

5 listexi.sml の実装

listexi 関数は，整数とリストを引数に取り，その整数がリスト中に何回出現するかをカウントして返す関数である．パターンマッチングを用いて，リストの構造に応じた再帰処理を行うことで実現している．

まずリストが空である場合（終了条件）には 0 を返す．それ以外の場合には，リストの先頭要素と比較し，一致していれば 1 を加算し，再帰的に残りのリストで同様の判定を行う．この処理により，リスト中の該当要素の合計出現回数を求めることができる．

以下にテスト入力とその実行結果を示す．

```
    val test1 = listexi(1, [1,2,3])
    val test2 = listexi(0, [1,2,3])
    val test3 = listexi(1, [1,2,1])
    val test4 = listexi(1, [])
```

実行結果は次のようになった.

```
    L$ sml listexi.sml
Standard ML of New Jersey v110.79 [built: Sat Oct 26 12:27:04 2019]
[opening listexi.sml]
listexi.sml:4.27 Warning: calling polyEqual
val listexi = fn : ''a * ''a list -> int
val test1 = 1 : int
val test2 = 0 : int
val test3 = 2 : int
val test4 = 0 : int
```

実行結果より，listexi 関数は指定した要素の出現回数を正しくカウントしていることが確認できる．この関数はリスト探索と条件分岐，再帰処理の組み合わせにより実装されており，基本的なリスト操作の理解に適した例である．作成した SML ソースコードは第 9.4 章に添付している.

6 listlen.sml

7 listsort.sml

7.1 listsplit.sml

7.2 マージソートの実装

8 SML や講義に関する所感

9 作成したプログラムのソースコード

9.1 listcat.sml のソースコード

```
1: fun listcat([], x) = x : int list
2:     |listcat(x, []) = x : int list
3:     |listcat(x::xs, y) = x :: listcat(xs, y)
```

9.2 listcmp.sml のソースコード

```
1: fun listcmp([],[]) = 0
2:     |listcmp([], x::xs) = x + listcmp([], xs)
3:     |listcmp(x::xs ,[]) = x + listcmp(xs, [])
4:     |listcmp(x::xs, y::ys) =
5:      let
6:          val diff = x - y
7:      in
```

```
 8:         diff + listcmp(xs, ys)
 9:     end;
```

9.3 listcpy.sml のソースコード

```
1: fun listcpy [] = []
2:    |listcpy (x::xs) = x :: listcpy (xs);
```

9.4 listexi.sml のソースコード

```
1: fun listexi(x, []) = 0
2:    |listexi(x, y::ys) =
3:     let
4:         val check = if (x = y) then 1 else 0
5:     in
6:         check + listexi(x, ys)
7:     end;
```

9.5 listlen.sml のソースコード

```
1: fun listlen([]) = 0
2:    |listlen(x::xs) = 1 + listlen(xs)
3: val test1 = listlen([])
4: val test2 = listlen([1])
5: val test3 = listlen([1, 2])
6: val test4 = listlen([1,2,3])
```

9.6 listsort.sml のソースコード

```
 1: fun merge([], ys) = ys
 2:   | merge(xs, []) = xs
 3:   | merge(x::xs, y::ys) =
 4:       if x <= y then x :: merge(xs, y::ys)
 5:                 else y :: merge(x::xs, ys);
 6:
 7: fun listsplit lst =
 8:   let
 9:     fun loop([], left, right) = (rev left, rev right)
10:       | loop([x], left, right) = (rev (x::left), rev right)
11:       | loop(x::y::rest, left, right) = loop(rest, x::left, y::right)
12:   in
13:     loop(lst, [], [])
14:   end;
15:
16: fun listsort([]) = []
17:    |listsort([x]) = [x]
18:    |listsort L =
19:     let
20:         val (left, right) = listsplit L
21:     in
22:         merge(listsort left, listsort right)
23:     end;
24: val test1 = listsort[5,4,3,2,1]
25: val test2 = listsort[2,3,1,2]
26: val test3 = listsort[]
27: val test4 = listsort[4,3,2,1]
```

9.7 listsplit.sml のソースコード

```
 1: fun listsplit lst =
 2:   let
 3:     fun loop([], left, right) = (rev left, rev right)
 4:       | loop([x], left, right) = (rev (x::left), rev right)
 5:       | loop(x::y::rest, left, right) = loop(rest, x::left, y::right)
 6:   in
 7:     loop(lst, [], [])
 8:   end;
 9:
10:
11: val test1 = listsplit([1,2,3,4])
12: val test2 = listsplit([1,2,3,4,5])
13: val test3 = listsplit([])
14: val test4 = listsplit([1])
```