COMPUTER SCIENCE TRIPOS Part IA – 2023 – Paper 1

1 Foundations of Computer Science (avsm2)

Given the following unsorted lists of values, we need to sort them by the frequency with which individual values occur within the list.

```
let input1 = [4; 1; 3; 3; 2; 3; 1]
let input2 = ['a'; 'e'; 'i'; 'e'; 'o'; 'e'; 'i']
```

We will use a run-length encoding (RLE) to encode repeated elements in a more compact form. The RLE representation has type `('a * int) list`, where the `int` is the number of times the `'a` value is repeated.

(*a*) Define a `rev` function which can reverse an input list in $O(n)$ time complexity. [2 marks]

(*b*) Define a `sort` function for a list, using an algorithm of your choice. The first argument to `sort` is a comparator function with arguments `a` and `b` that returns `0` if `(a=b)`, a negative value if `(a<b)`, and a positive value if `(a>b)`.

```
val sort : ('a -> 'a -> int) -> 'a list -> 'a list
```

[6 marks]

(*c*) Define `rle_encode` that converts a sorted list of elements into an RLE encoded version, and `rle_decode` that converts them back to the original list.

```
val rle_encode : 'a list -> ('a * int) list
val rle_decode : ('a * int) list -> 'a list
```

[6 marks]

(*d*) Assume the comparator functions `int_cmp` and `char_cmp` are already defined:

```
val int_cmp : int -> int -> int
val char_cmp : char -> char -> int
```

Using all the earlier functions, define `freq_sort` that sorts the elements in ascending order of the total number of occurrences of each element within the list. When applied to the lists defined earlier, it should output:

```
# freq_sort int_cmp input1;;
- : int list = [2; 4; 1; 1; 3; 3; 3]
# freq_sort char_cmp input2;;
- : char list = ['a'; 'o'; 'i'; 'i'; 'e'; 'e'; 'e']
```

[6 marks]