List and Recursion

CS316 2014/15 Practical 3

Deadline: Monday 27 October.

The practical will be marked in the supervised laboratory session on Monday 27 October. Consequently, you should prepare for the laboratory session by attempting these problems in advance. Remember, half of the marks for each question will be awarded for the correctness of your solutions and half for a *short* written explanation describing why your answer is correct. Emphasis is placed on the word *short* — we are not looking for essays, just enough information to demonstrate that you understand your code.

The files that you need for this exercise are available on the Web - download each of the files for this practical to your account. Edit the file Prac3.hs to contain your answers to the questions. Remember to write your name and user id at the top of the file.

Question 1: Use recursion to define a function `factorial` which takes as input a non-negative integer and returns the result of multiplying all the numbers from 1 to the input. For example

```
factorial 5 = 1 * 2 * 3 * 4 * 5 = 120
```

An error message should be returned for negative input.

Question 2: Use recursion to define a function `position` which has as input an integer, a character and a string and returns the result of inserting the character in the string at the position specified by the integer. For example

```
position 5 'a' ``dgh*sja'' = ``dgh*saja''
```

An error message should be returned for negative input or if the position is more than the length of the list. Define a function `position2` which has the same effect as `position` but which is defined using `take` and `drop` instead of recursion.

Question 3: Use list recursion to define the function `mySum` which takes as input an integer and a list of integers and returns the list obtained by adding every element of the list by the first input. For example

```
mySum 3 [1,5,3,2,-1] = [4,8,6,5,2]
```

Define a function `mySum2` which has the same effect as `mySum` but which is defined using list comprehension instead of recursion.

Question 4: In this question we will define a function for sorting lists based upon the algorithm *bucket sort.*

- First, define a function `smallest` which takes as input a list of integers and returns the smallest element in the list
- Next, define a function `delete` which takes as input an integer and a list of integers and returns the list obtained by deleting the first occurrence of the integer in the list
- Finally define the function `bucket` which takes a list of integers and returns the list whose head is the smallest element in the list and whose tail is the result of recursively sorting the list obtained by deleting the smallest element of the list from the list.

Question 5: Use recursion to define a function `copy` has as input an integer and and element of a type and returns a list whose length is the first input and whose elements are the second input. For example

```
copy 5 'a' = ['a', 'a', 'a', 'a', 'a']
```

An error message should be returned for negative input. Define a function `copy2` which has the same effect as `copy` but which is defined using list comprehension and not recursion. [*Hint: Recall that* `[1 .. n]` *denoted the list of numbers from* `1` *to* `n`]

Question 6: Here are three algorithms for reversing lists

- To reverse a list, append the head of the list to the end of the result of reversing the tail of the list
- To reverse a list, split the list in half, reverse each half, and append the reversed second half of the list to front of the reversed first half of the list.
- Define a function which takes two lists as inputs and adds each element of the first list onto the front of the second list. Then, to reverse a list, call the first function with the the first input being the list to be reversed and the second list being the empty list.

Define functions `rev1, rev2` and `rev3` which implement these algorithms. For the second algorithm you may wish to use the functions `take, drop`.