HW1 - Higher order functions and Typing

PPL 2018 - http://www.cs.bgu.ac.il/~ppl182 (http://www.cs.bgu.ac.il/~ppl182)

Q1 Theoretical Questions (24 points)

Q1.1 Type Intersection (6 points)

What is the intersection of the types denoted by the expressions T1 and T2? Describe values in the type (give examples, characterize them) and write a TypeScript type expression that denotes this intersection.

```
1.1.1 T1 = {a:number[]; }   T2 = {b:string};

1.1.2 T1 = {a: {b: number}} T2 = {a: {c: string}};

1.1.3 T1 = {a: number[]; }  T2 = {a: number};
```

Q1.2 Type Inclusion (6 points)

Are the following types T1 and T2 subsets one of another? Answer T1 < T2, or T2 < T1 or none. Justify your answer.

```
1.2.1
    type T1 = {a:number, b:{}}[]
    type T2 = {a:number}[]

1.2.2
    type T1 = {a: {c: any}, b:any}
    type T2 = {a: {c: number}, b: number}

1.2.3
    type T1 = {a:number, b:undefined}
    type T2 = {a:number, b:any}
```

Q1.3 Type Inference (8 points)

Write the most specific TypeScript type expression associated to each of the following expressions:

```
1.3.1 let v1 = { name:"peter", age:20 };

1.3.2 v2 = { children: [ {name: "john"}, {age:12} ] };

1.3.3 v3 = (x) => x + 2;

1.3.4 v4 = (f, l) => map((x)=>f(f(x)), l);
```

Q1.4 Type Definitions (4 points)

1.4.1 Is it possible to define a type in TypeScript for the set of all strings with length larger than 2 using the type constructors defined in class?

1.4.2 Is it possible to define a type for the set of all numbers larger than 0?

Justify your answers.

Q2 TypeScript Programming (36 points)

Q2.1 BinTree (12 points)

Consider the definition of the recursive type BinTree that we saw in class:

```
In [8]: interface BinTree {
            root: number;
            left?: BinTree;
            right?: BinTree;
        };
```

```
Out[8]: undefined
```

Q.2.1.1 TreePreArray

Implement the function `TreePreArray` which gets a `BinTree` as argument and returns an array with the values of the nodes in Pre-order.

Specify the type of the function, write its definition, and provide at least 3 tests (using assert).

In [9]:
```
const TreePreArray //
```
Out[9]:
```
undefined
```

Q.2.1.2 TreeInArray

Implement the function `TreeInArray` which gets a `BinTree` as argument and returns an array with the values of the nodes in In-order.

Specify the type of the function, write its definition, and provide at least 3 tests (using assert).

In [11]:
```
const TreeInArray //
```
Out[11]:
```
undefined
```

Q2.1.3 TreePostArray

Implement the function `TreePostArray` which gets a `BinTree` as argument and returns an array with the values of the nodes in Post-order.

Specify the type of the function, write its definition, and provide at least 3 tests (using assert).

In [12]:
```
const TreePostArray //
```
Out[12]:
```
undefined
```

Consider the definition of the recursive type declaration of the generic GBinTree that we saw in class:

In [13]:
```
interface GBinTree<T> {
    root: T;
    left?: GBinTree<T>;
    right?: GBinTree<T>;
};
```
Out[13]:
```
undefined
```

Repeat Q2.1.1, Q2.1.2, Q2.1.3 for the generic GBinTree:

Q2.1.4 GBinTreePreArray

In [15]:
```
const GBinTreePreArray // @@@
```
Out[15]:
```
undefined
```

Q2.1.5 GBinTreeInArray

```
In [16]: const GBinTreeInArray // @@@
Out[16]: undefined
```

Q2.1.6 GBinTreePostArray

```
In [17]: const GBinTreePostArray // @@@
Out[17]: undefined
```

Q2.2 Subsets (14 points)

Q2.2.1 KSubsets

Implement the function KSubsets which gets an array A of n elements and a number k as input, and returns an array with all the subsets of size k of A (each subset will be an array itself).

Define the type of the function, and implement at least 3 tests for the function (using `assert`).

```
In [20]: const KSubsets // Define the type and the function
Out[20]: true
```

For example: KSubsets ([1,2,3],2) => [[1,2],[1,3],[2,3]]

Q2.2.2 AllSubsets

Implement the function `AllSubsets` which gets an array A of n elements and returns an array with all the subsets of A (each subset will be an array itself).

Provide the type of the function, its definition and at least 3 examples (using `assert`).

```
In [18]: const AllSubsets // <define the type and the function>
Out[18]: undefined
```

For example: AllSubsets([1,2,3]) => [[], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]]

Q2.3 Flatmap (10 points)

Q2.3.1 Flatmap Definition

Implement the function `flatmap` which gets a function `f` and an array `A` as parameter, and returns the concatenation of all the arrays returned by `f` when it is applied to each element in `A`.

Provide the generic type for `flatmap`, the function definition and at least 3 examples testing its execution using `assert`.

```
In [23]: const flatmap; //
Out[23]: undefined
```

For example: Flatmap((x)=>x[0], [[[1,2], [3,4]], [[5,6], [7,8]]]) => [1,2,5,6]

Q2.3.2 Using Flatmap

Consider the movie list (similar to the values we manipulated in practice session):

In [24]:
```
let movieLists = [
            {
                name: "Instant Queue",
                videos : [
                    {
                        "id": 70111470,
                        "title": "Die Hard",
                        "boxarts": [
                            { width: 150, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/DieHard150.jpg" },
                            { width: 200, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/DieHard200.jpg" }
                        ],
                        "url": "http://api.netflix.com/catalog/titles/movies/7
0111470",
                        "rating": 4.0,
                        "bookmark": []
                    },
                    {
                        "id": 654356453,
                        "title": "Bad Boys",
                        "boxarts": [
                            { width: 200, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/BadBoys200.jpg" },
                            { width: 150, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/BadBoys150.jpg" }

                        ],
                        "url": "http://api.netflix.com/catalog/titles/movies/7
0111470",
                        "rating": 5.0,
                        "bookmark": [{ id: 432534, time: 65876586 }]
                    }
                ]
            },
            {
                name: "New Releases",
                videos: [
                    {
                        "id": 65432445,
                        "title": "The Chamber",
                        "boxarts": [
                            { width: 150, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/TheChamber150.jpg" },
                            { width: 200, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/TheChamber200.jpg" }
                        ],
                        "url": "http://api.netflix.com/catalog/titles/movies/7
0111470",
                        "rating": 4.0,
                        "bookmark": []
                    },
                    {
                        "id": 675465,
                        "title": "Fracture",
                        "boxarts": [
                            { width: 200, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/Fracture200.jpg" },
                            { width: 150, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/Fracture150.jpg" },
                            { width: 300, height: 200, url: "http://cdn-0.nflx
img.com/images/2891/Fracture300.jpg" }
                        ],
                        "url": "http://api.netflix.com/catalog/titles/movies/7
0111470",
                        "rating": 5.0,
                        "bookmark": [{ id: 432534, time: 65876586 }]
                    }
                ]
            }
        ]
```

```
Out[24]: undefined
```

Implement a function `getBoxArts()` which gets a type similar to `movieLists` and returns a map `{id, title, boxart}` for every video in the movieLists, such that the `boxart` property in the result will be the url of the boxart object with dimensions of 150x200px.

Your implementation must use `map(), flatmap()`, and `filter()`.

You are **not allowed** to use indexers - which means any expression such as `movieLists[i]` and any **mutation**.

Provide the exact type of the function (as specific as possible), its definition and at least 3 tests using `assert`.

```
In [25]: const getBoxarts //
```

```
Out[25]: undefined
```

Q3 Scheme Programing (40 points)

Q3.1 count-syllables (10 points)

Implement the procedure `count-syllables` which takes a list of letters as its argument and returns the number of syllables in the word formed by the letters, according to the following rule:

The number of syllables is the number of vowels, except that a group of consecutive vowels counts as one. Vowels are the letters:

```
(define vowels '(a e i o u))
```

For example: in the word “soaring,” the group “oa” represents one syllable and the vowel “i” represents a second one.

Provide at least 5 test cases that expose likely failures of your procedure. For example: word ends with a vowel, ends with two vowels in a row. more than two consecutive vowels.

Your procedure must not use any mutation and must not use any direct indexing inside the list parameter.

Example:

```
> (count-syllables '(s o a r i n g))
2

> (count-syllables '(b e e p))
1
```

Q3.2 sorted? (10 points)

Implement the procedure `sorted?` which takes as input a list of numbers and returns true if the list is sorted according to the comparator argument.

Example:

```
> (sorted? '(1 3 5) <)
==> #t
```

What should the procedure return on an empty list?
On a list of 1 element? Justify.

Q3.3 merge (10 points)

Implement the procedure `merge` which takes two lists of numbers as input. Each list must be a sorted list of numbers in increasing order. Merge must return a list containing all of the numbers, in increasing order.

Example:

```
> (merge '(1 3 8) '(2 5 6))
==> '(1 2 3 5 6 8)
```

Define the pre-conditions of the procedure. The procedure `merge` must test its pre-conditions and throw an error if they are not met.

Provide the function definition and at least 3 examples.

Q3.4 remove-adjacent-duplicates (10 points)

Implement the procedure `remove-adjacent-duplicates` which takes a list as input and returns as value the same list with any sequence of repeated elements reduced to a single element:

Example:

```
(remove-adjacent-duplicates ’(y a b b a d a b b a d o o))
'(y a b a d a b a d o)

(remove-adjacent-duplicates ’(yeah yeah yeah))
'(yeah)
```

Provide the function definition and at least 3 examples.

בהצלחה

```
In [ ]:
```