Answers for theoretical questions

1.1:

1.1.1: the intersaction of the types denoted by T1 and T2 is {a:number[],b:string}. It is map that contain the keys 'a' and 'b'. the type we can put at 'a' key is array of numbers, at key 'b' we can put string type. Examples: 1) {a:[1], b:"one"}

2) {a:[1,2], b:"one and two"}

1.1.2: the intersaction of the types denoted by T1 and T2 is

{a: {b:number, c:string}}. it is a map that contains

the key 'a'. the type we can put at 'a' key is map that contains the keys 'b' and 'c'. the type we can put

at 'b' is number, and at 'c' we can put string.

examples: 1) {a: {b: 1, c: "one"}}

2) {a: {b: 2,c: {"two"}}

1.1.3: the intersaction of the types denoted by T1 and T2 is

{a: number[] & number} or {a: undefined|null}.

A value that satisfies the type now has to be a number and an array containing numbers. As there is no way that a number is something different than number and definetly not an array, nothing can satisfy this type.examples: 1) undefined

2) {a: 1}

1.2:

1.2.1: T1<T2. T1 type is a subtype of T2. T1 is an array that contains map, that map contains keys 'a' and 'b'.

'a' values will be from the type of number and 'b' values will be map. T2 is also an array that contains map.

T2 is an array that contains map. that map contains key 'a'.

'a' values will be from the type of number.

T1 is subtype of T2 because it contains at any objects at it's array a map that contains key 'a' as T2 do,

but also have limitations on key 'b' which has to be map, while at T2 there is no any limitation about key 'b'.

1.2.2: T2<T1. T2 type is a subtype of T1. T1 is a map that contains keys 'a' and 'b'.

'a' value will be from the type of map. that map contains key 'c' that its value will be from any type. 'b' value will be from any

type. T2 is also a map that contains keys 'a' and 'b'. 'a' value will be from the type of map.

That map contains key 'c' that its value will be from type of number. 'b' value will be from type number. as we can see,

T1 and T2 are both maps that contains keys 'a' and 'b' but T2 is more specific about 'a' and 'b' values.

while T1 can get any type at 'b' key, for example, T2 can get type of numbers. therefore, T2 can be subtype of T1.

1.2.3: T1<T2. T1 type is a subtype of T2. T1 is a map that contains keys 'a' and 'b'. 'a' value will be from the type of number. 'b' value will be from the type of type undefined. T2 is also a map that contains keys 'a' and 'b'. 'a' value can be from the type of number, as well as T1 'a' key is. but 'b' key can be from any type, while at T1, 'b' value can be only from type undefined. therefore, T1 is more specific about 'b' value. therefore, T1 can be subtype of T2.

1.3:

1.3.1: type of v1 is: {name:string, age:number}.

1.3.2: type of v2 is: {children: {name:string, age:number}[]}

1.3.3: type of v3 is: (x:number)=> number

1.3.4: type of v4 is: (f:(T)=>R,l:T[])=>map:(f:(R)=>E,arr:R[])=>E[]

1.4:

1.4.1: No. As we learned in class,types are sets of values. we can not define a specific type. we can apply operations on sets

1.4.2: No. As we learned in class,types are sets of values. we can not define a specific type. we can apply operations on sets