תרגיל 2 – תיכון תוכנה – יבש

1.

class Observable<T> {  
 private val listeners = HashSet<(T) -> Unit>()  
 fun listen(listener: (T) -> Unit): Unit {  
 listeners.add(listener)  
 }  
 fun unlisten(listener: (T) -> Unit): Unit {  
 listeners.remove(listener)  
 }  
 protected fun onChange(t: T): Unit {  
 listeners.forEach **{** it(t) **}** }  
 companion object {  
 fun <T> of(t: T) = object : Observable<T> {  
 override fun listen(f: T -> Unit) = f(t)  
 }  
 }  
 fun <S> flatMap(f: (T) -> Observable<S>): Observable<S>{  
 val res = Observable<S>()  
 res.listen **{** f(it).onChange() **}** return res  
 }  
}

2.

צ"ל: Observable<T>.of(t).flatmap(f) == f(t)

הוכחה:

Observable<T>.of(t) returns an object which is an Observable with listen that returns f(t). Therefore, Observable<T>.of(t).flatmap(f) will call **this** listen which returns f(t) (as an Observable).

צ"ל: o.flatMap{ Observable<T>.of(it) } == o

הוכחה:

o.flatMap{ Observable<T>.of(it)} is calling listen with Observable<T>.of(it), which is just like the normal listen of o. Therefore o.flatMap{ Observable<T>.of(it) } == o.

צ"ל: o.flatMap(f).flatMap(g) === o.flatMap{ f(it).flatMap(g) }

הוכחה:

o.flatmap(f) will call listen with f. Using flatmap(g) then will call listen with g then.  
o.flatMap{ f(it).flatMap(g) } will call listen with f(it).flatmap(g), which is calling listen with f, and then calling listen with g, as requested.