Lista 5

Ewaluacja gorliwa i leniwa

W poniższych zadaniach **dopuszczalne jest** wykorzystanie funkcji wbudowanych obliczających **długość listy**, **odwracających listę** oraz **łączących dwie listy**, o ile **nie wpływają one na drastyczne pogorszenie złożoności obliczeniowej**.

Każde zadanie, poza implementacją funkcji, musi posiadać **kompletny zestaw testów**.

Do wykonania zadań należy wykorzystać mechanizmy poznane na wykładzie nr 5.

1) Wykorzystując **abstrakcję funkcyjną** zamodeluj poniższą sytuację:
„Na stole ustawiono w rzędzie losy na loterię. Każdy z losów można odkryć, zyskując w ten sposób ukrytą pod nim nagrodę. Losy mogą kryć w sobie nagrody lub kolejne losy. Próba odkrycia odkrytego losu nie zmienia stanu na stole.”

   Potraktuj stół jako zwykłą listę zawierającą losy. Zdefiniuj funkcję *buyTicket*, która dla danego stołu oraz numeru losu dokonuje jego odsłonięcia (zwracając nową listę). (OCaml) (30 pkt.)

   Przykładowo:
[?; ?; ?; ?; ?] ---- 3 ----> [?; ?; „Komputer”; ?; ?] ---- 4 ----> [?; ?; „Komputer”; ?; ?],
gdzie „?” oznacza nieodkryty los, a „?” – los ukryty pod oryginalnym losem.

   UWAGA: Zwróć uwagę, że powyższy przykład **nie** przedstawia dokładnie tego, co interpreter będzie wyświetlał na ekranie. Stwierdzenie, że „**los jest nieodkryty**” oznacza, że jego wartość **nie** została jeszcze **wyliczona**. Losy mogą zawierać **nagrody dowolnego rodzaju** tj. nie tylko łańcuchy jak w powyższym przykładzie.

2) Zdefiniuj funkcję *skipTakeL*, która dla danej listy leniwej zwraca nową listę leniwą zbudowaną z drugiego elementu, elementu leżącego w odległości 2 od poprzedniego, z elementu leżącego w odległości 4 od poprzedniego, w odległości 6 itd. (Scala) (20 pkt.)

   Przykładowo:
$[x_1; x_2; x_3; x_4; x_5; x_6; x_7; ...]$ --------> $[x_2; x_4; x_8; x_{14}; x_{20}; ...]$