Консультация к коллоквиуму по курсу «Формальная спецификация и верификация программ. Часть I. Спецификация программ» 20.10.2010

Билет № 1 ФИО:_____, группа _____, бонус___

Дать эквивалентное implicit-определение функции **variable** x : **Int** **value** f: **Int** -> **write any Unit** f(t) **is** (**if** x = t **then** x := t **end**)	Ответ:

---

1. «Неазартное 21» : Задача системы - визуализировать поле игры (Game) и модифициро-вать его в ответ на команды игрока. В колоде 36 карт (вальты, дамы, короли, тузы, 6-ки, 7-ки, 8-ки, 9-ки и 10-ки четырех мастей); стоимость карт: валет - 2, дама - 3, король – 4, туз – 11, 6-ки–10-ки имеют стоимость, соответствующую своему названию. Банкир (Banker) играет с единственным игроком-соперником (Gambler). Сначала игрок берет из колоды (PackOfCards) себе карты (набирает свой Hand), затем банкир берет из колоды себе карты. Если сумма набранных баллов по картам у игрока равна 21, он выиграл; больше 21 – проиграл; иначе если у банкира 21, он выиграл; если больше 21 – проиграл; в противном случае выигрывает тот, кто набрал б*о*льшую сумму баллов.

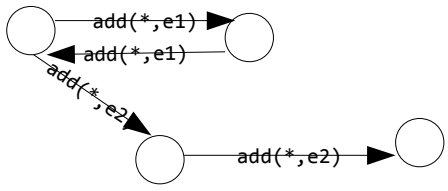
```
type Token, Card = Token -m-> {| n : Nat :- n isin {2..11}\{5} |},
     Hand = Card-set, Banker = Hand, Gambler = Hand,
     PackOfCards = Card-list, Game = Gambler >< PackOfCards >< Banker
value newgame : Unit -~-> Game,
             -- если карта берется, то в случае gamblerMove она переносится
             -- из PackOfCards в Gambler, в случае bankerMove – в Banker
gamblerMove : Game >< Bool -~-> Game, -- 2-й аргумент-прекращение выбора карт
bankerMove : Game >< Bool -~-> Game, -- 2-й аргумент-прекращение выбора карт
```

2. Определить, противоречива ли данная алгебраическая спецификация. Ответ обосновать (доказать противоречие или привести модель спецификации в доказательство непротиворечивости). Считать, что во всех типах есть достаточное количество различных значений.

```
type Place, Name, Volume, Info, DB
value   empty: DB,
   free: DB >< Place -> Bool,
   make: Name >< Volume -> Info,
   name: Info -> Name,
   volume: Info -> Volume,
   add: DB >< Place >< Info -~-> DB,
   get: DB >< Place -~-> Info,
   update: DB >< Place >< Name -~-> DB
axiom forall db: DB, pi, p: Place, info:
          Info, n: Name, v: Volume :-
   free(empty, p),
   free(add(db, pi, _), p) is
        pi ~= p /\ free(db, p),
   name(make(n, _)) is n,
   volume(make(_, v)) is v,
   get(add(db, pi, info), p) is
     if pi = p then info else get(db, p) end,
   update(add(db, p, info), p, n) is
      add(db, p, make(n, volume(info))),
   update(add(db, pi, info), p, n) is
      add(update(db, p, n), pi, info)
```

3. Существует ли модель, отвечающая приведенной спецификации, чей конечный автомат содержал бы представленный в задании подавтомат? Ответ обосновать (доказать несуществование или привести модель в доказательство существования).

```
type E, S
value empty: S, add: S >< E -> S,
  tail: S -~-> S,
  include: S >< E -> Bool
axiom forall s: S, e, e1: E :-
  tail(add(s,e)) is s,
  include(add(s,e1),e) is e = e1 \/ include(s,e),
  ~include(empty,e)
```



Примечание: **оформлять ответ к задаче №1 на отдельном листе**