Facit och kommentarer till tentamen 2021-03-15 i DA4003

Del A: kodfrågor

1. **Imperativ programmering:**

   a) Möjlig lösning:

```
void matrix_encrypt(char* s, int n, int cols)
{
  // calculate how many rows the matrix should have
  // we have to take into account if cols divides n or not
  int rows = n % cols == 0 ? n / cols : n / cols + 1;

  for(int i = 0; i < cols; i++)
    for(int j = 0; j < rows; j++) {
      // calculate the index to fetch
      int k = j * cols + i;

      // we should only fetch the index if it's not out of bounds
      if (k < n)
        printf("%c",s[k]);
    }

  printf("\n");
}
```

   b) Med varianten av `matrix_encrypt` ovan blir detta ganska trivialt. Hade vi haft några lokala arrayer hade detta varit lite mer komplicerat.

```
void matrix_encrypt_pointer(char* s, int n, int cols)
{
  // calculate how many rows the matrix should have
  // we have to take into account if cols divides n or not
  int rows = n % cols == 0 ? n / cols : n / cols + 1;

  for(int i = 0; i < cols; i++)
    for(int j = 0; j < rows; j++) {
      // calculate the index to fetch
      int k = j * cols + i;

      // we should only fetch the index if it's not out of bounds
      if (k < n)
        printf("%c",*(s + k));
    }

  printf("\n");
}
```

2. **Objektorienterad programmering:**

   a) Möjlig lösning:

```
class Dice {

    private int number_of_dice;
    private int number_of_sides;

    public Dice(int number_of_dice, int number_of_sides) {

        if (number_of_sides == 4 || number_of_sides == 6  ||
            number_of_sides == 8 || number_of_sides == 12 ||
```

```
            number_of_sides == 20) {
            this.number_of_dice = number_of_dice;
            this.number_of_sides = number_of_sides;
        } else {
            throw new IllegalArgumentException("Incorrect number of side");
        }
    }

    public int get_number_of_dice() {
        return number_of_dice;
    }

    public int get_number_of_sides() {
        return number_of_sides;
    }

    public int roll_dice() {
        Random r = new Random();

        int sum = 0;

        for (int i = 0; i < number_of_dice; i++)
            sum += r.nextInt(number_of_sides) + 1;

        return sum;
    }

    public int roll_dice_many_times(int n) {
        int sum = 0;

        for (int i = 0; i < n; i++)
            sum += roll_dice();

        return sum;
    }
}
```

b) Möjlig lösning:

```
class Dice6 extends Dice {

    public Dice6() {
        super(1,6);
    }
}
```

3. Webb och händelsestyrd programmering:

a) Möjlig lösning:

```
function is_complete(g) {

    var out = true;

    for (var i = 1; i < 4; i++)
        for (var j = 1; j < 4; j++) {
            out &&= g[i][j] == 0;
        }

    return out;
}
```

b) Möjlig lösning:

```
function best_value(g) {
    var bestval = 0;

    for (var i = 1; i < 4; i++) {
        for (var j = 1; j < 4; j++) {
            s = sum_neighbors(g,i,j);
            if (s > bestval && g[i][j] != 0)
                bestval = s;
        }
    }

    return bestval;
}
```

c) Möjlig lösning:

```
function klick() {
    var cell = this;
    var g = get_grid();

    sum = sum_neighbors(g,parseInt(cell.id[1]),parseInt(cell.id[3]));
    best = best_value(g);

    if (sum != best) {
        alert("You can do better!");
    } else {
        cell.innerHTML = "0";
        cell.style.backgroundColor = "#b6d9ee";
        cell.onclick = null;
    }

    if (is_complete(get_grid()))
        alert("Congratulations!");

}
```

4. **Funktionell programmering:** Möjlig lösning:[1]

```
data IntExpr = Const Int
             | Var String
             | Add IntExpr IntExpr
             | Mul IntExpr IntExpr
  deriving (Eq,Show)

data BoolExpr = IntEq IntExpr IntExpr
              | BoolNot BoolExpr
              -- a
              | Or BoolExpr BoolExpr
              | And BoolExpr BoolExpr
              -- b
              | IntGT IntExpr IntExpr
              | IntLT IntExpr IntExpr
  deriving (Eq,Show)

evalIntExpr :: IntExpr -> [(String,Int)] -> Int
evalIntExpr = undefined

evalBoolExpr :: BoolExpr -> [(String,Int)] -> Bool
```

[1] Vissa funtioner och fall är `undefined` då de frågas efter i labben.

```
evalBoolExpr (IntEq e1 e2) env = undefined
evalBoolExpr (BoolNot x) env = undefined
-- a
evalBoolExpr (And b1 b2) env = evalBoolExpr b1 env && evalBoolExpr b2 env
evalBoolExpr (Or b1 b2) env = evalBoolExpr b1 env || evalBoolExpr b2 env
-- b
evalBoolExpr (IntGT e1 e2) env = evalIntExpr e1 env > evalIntExpr e2 env
evalBoolExpr (IntLT e1 e2) env = evalIntExpr e1 env < evalIntExpr e2 env

-- c
geq :: IntExpr -> IntExpr -> BoolExpr
geq e1 e2 = Or (IntGT e1 e2) (IntEq e1 e2)

leq :: IntExpr -> IntExpr -> BoolExpr
leq e1 e2 = Or (IntLT e1 e2) (IntEq e1 e2)

-- d
data Instr = Assign String IntExpr
           | While BoolExpr [Instr]
  deriving (Eq,Show)

for :: (String,IntExpr) -> BoolExpr -> Instr -> [Instr] -> [Instr]
for (x,e) cond upd body = [Assign x e,While cond (body ++ [upd])]
```

5. **Logikprogrammering:**

   a) Möjlig lösning:

   ```
   nub([],[]).
   nub([X|XS],YS) :- member(X,XS), nub(XS,YS).
   nub([X|XS],[X|YS]) :- nub(XS,YS).
   ```

   b) Möjlig lösning:

   ```
   compress([],[]).
   compress([X],[(X,1)]).
   compress([X,Y|XS],[(X,1)|YS]) :- X \= Y, compress([Y|XS],YS).
   compress([X,X|XS],[(X,N)|YS]) :- compress([X|XS],[(X,M)|YS]), N is 1 + M.
   ```

   c) Möjlig lösning:

   ```
   replicate(0,_,[]).
   replicate(N,X,[X|XS]) :-
       M is N - 1,
       replicate(M,X,XS).

   uncompress([],[]).
   uncompress([(X,N)|XS],YS) :-
       replicate(N,X,ZS),
       uncompress(XS,WS),
       append(ZS,WS,YS).
   ```

Del B: flervalsfrågor (1p per fråga)

6. *A,D*

7. *C*

8. *D*

9. *B*

10. *E*

11. *D*

12. *D*

13. *B,E*