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

Del A: kodfrågor

1. **Imperativ programmering:**

a) Möjlig lösning:

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

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

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

b) Möjlig lösning:

```
void matrix_decrypt_goto(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;

  int i = 0;

 startouter: ;

  int j = 0;

 startinner: ;

  // calculate the index to fetch
  int k = j * rows + i;

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

  j++;
  if (j < cols)
    goto startinner;

  i++;
  if (i < rows)
    goto startouter;

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

2. **Objektorienterad programmering:**

a) Möjlig lösning:

```
import java.util.Map;
import java.util.HashMap;

abstract class AlgExpr {
    abstract Integer eval(Map<Character,Integer> d);
}

class Constant extends AlgExpr {

    private Integer value;

    public Constant(Integer value) {
        this.value = value;
    }

    public String toString() {
        return value.toString();
    }

    public Integer eval(Map<Character,Integer> d) {
        return value;
    }
}

class Var extends AlgExpr {
    private Character var;

    public Var(Character var) {
        this.var = var;
    }

    public Integer eval(Map<Character,Integer> d) {
        return d.get(var);
    }

    public String toString() {
        return var.toString();
    }
}

class Add extends AlgExpr {
    private AlgExpr e1;
    private AlgExpr e2;

    public Add(AlgExpr e1, AlgExpr e2) {
        this.e1 = e1;
        this.e2 = e2;
    }

    public String toString() {
        return e1.toString() + " + " + e2.toString();
    }

    public Integer eval(Map<Character,Integer> d) {
        return e1.eval(d) + e2.eval(d);
    }
}
```

3. Webb och händelsestyrd programmering:

a) Möjlig lösning:

```
function init(g) {

    for (var i = 0; i < 5; i++) {
        for (var j = 0; j < 5; j++) {
            var tile = document.getElementById("c" + i + "x" + j);

            if (g[i][j] == 1)
                tile.onclick = klick;

            if (g[i][j] == 0)
                tile.style.backgroundColor = "#b6d9ee";

            tile.innerHTML = g[i][j];
        }
    }
}
```

b) Möjlig lösning:

```
function sum_row_col(g,r,c) {

    var sum = 0;

    for (var i = 0; i < 5; i++)
        if (i != c)
            sum += g[r][i];

    for (var i = 0; i < 5; i++)
        if (i != r)
            sum += g[i][c];

    return sum;
}
```

c) Möjlig lösning:

```
function is_complete(g) {

    var out = true;

    for (var i = 0; i < 5; i++)
        for (var j = 0; j < 5; j++) {
            out &&= g[i][j] == 0;
        }

    return out;
}
```

4. Funktionell programmering:

a) Möjlig lösning:

```
mask :: String -> String -> String
mask [] _ = []
mask s1 [] = s1
mask (x:s1) (y:s2) | x == y = '*' : mask s1 s2
                   | otherwise = x : mask s1 (y:s2)
```

b) Möjlig lösning:

```
mapFilter :: (a -> b) -> (b -> Bool) -> [a] -> [b]
mapFilter f p [] = []
mapFilter f p (x : xs) | p (f x) = f x : mapFilter f p xs
                       | otherwise = mapFilter f p xs
```

c) Möjlig lösning:

```
splitUp :: String -> Int -> [String]
splitUp [] _ = []
splitUp xs n = take n xs : splitUp (drop n xs) n

combine :: [String] -> String
combine xs | all (==[]) xs = ""
           | otherwise = concatMap (take 1) xs ++ combine (map (drop 1) xs)

matrixEncrypt :: String -> Int -> String
matrixEncrypt s cols = combine (splitUp s cols)
```

5. **Logikprogrammering:**

a) Möjlig lösning:

```
mask([],_,[]).
mask(XS,[],XS).
mask([X|XS],[X|YS],[42|ZS]) :- mask(XS,YS,ZS).
mask([X|XS],YS,[X|ZS]) :- mask(XS,YS,ZS).
```

b) Möjlig lösning:

```
dna_mass(S,M) :- string_to_list(S,XS), dna_mass_help(XS,M).

dna_mass_help([],0).
dna_mass_help([65|S],N) :- dna_mass_help(S,M), N is 313.2 + M.
dna_mass_help([67|S],N) :- dna_mass_help(S,M), N is 289.2 + M.
dna_mass_help([71|S],N) :- dna_mass_help(S,M), N is 329.2 + M.
dna_mass_help([84|S],N) :- dna_mass_help(S,M), N is 304.2 + M.
dna_mass_help([_|S],N) :- dna_mass_help(S,N).
```

Del B: flervalsfrågor (1p per fråga)

6. *B, E*
7. *D*
8. *A*
9. *C*
10. *A*
11. *C*
12. *D*
13. *A, D, E*