Facit och kommentarer till tentamen 2024-03-08 i DA4003

Del 1: flervalsfrågor (1p per fråga)

1. A, D, E
2. A
3. B
4. E
5. B, C, D
6. D
7. E
8. C

Del 2: kodfrågor

9. (a) Möjlig lösning:

```
int fib_rec(int n) {

  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (fib_rec(n-1) + fib_rec(n-2));

}
```

(b) Möjlig lösning:

```
int fib_while(int n) {

  int f0 = 0;
  int f1 = 1;

  int i = 0;
  int temp = 0;

  while (i < n) {
    temp = f0;
    f0 = f1;
    f1 += temp;
    i++;
  }

  return f0;
}
```

(c) Möjlig lösning:

```
int* fib_numbers(int n) {
  int* f = malloc(n * sizeof(int));

  // Not a very efficient solution:
  // for (int i = 0; i < n; i++)
  //   *(f + i) = fib_while(i);
  // return f;
```

```
  // Nicer solution:

  // Initialize two first values
  *f = 0;
  f++;
  *f = 1;
  f++;

  // Loop to get the rest
  for (int i = 0; i < n-2; i++) {
    *f = *(f - 1) + *(f - 2);
    f++;
  }

  // Reset the pointer
  return (f - n);
}
```

10. (a) Möjlig lösning:

```
class Rational {

    public Integer numerator;
    public Integer denominator;

    public Rational(Integer n,Integer d) {
        if (d == 0) {
            numerator = 0;
            denominator = 0;
        } else {
            numerator = n;
            denominator = d;
        }
    }
}
```

(b) Möjlig lösning:

```
    public String toString() {
        return (numerator.toString() + "/" + denominator.toString());
    }
```

(c) Möjlig lösning:

```
class NormalizedRational extends Rational {

    // Not needed for score
    private Integer gcd(Integer a, Integer b) {
        if (b == 0)
            return a;
        return gcd(b,a % b);
    }

    public NormalizedRational(Integer n,Integer d) {
        super(n,d);
        if (d != 0) {
            if (n == 0) {
                numerator = 0;
                denominator = 1;
            } else {
                numerator /= gcd(n,d);
```

```
                denominator /= gcd(n,d);
            }
        }
    }
}
```

11. (a) Möjlig lösning:

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

(b)

i. Möjlig lösning:

```
data Winner = Player Int | Draw
  deriving (Eq,Show)
```

ii. Möjlig lösning:

```
play :: (RPS,RPS) -> Winner
play (Paper,Rock)    = Player 1
play (Rock,Paper)    = Player 2
play (Rock,Scissor)  = Player 1
play (Scissor,Rock)  = Player 2
play (Scissor,Paper) = Player 1
play (Paper,Scissor) = Player 2
play _               = Draw
```

iii. Möjlig lösning:

```
winner :: [(RPS,RPS)] -> Winner
winner ps =
  let ws = map play ps
      p1 = length (filter (==Player 1) ws)
      p2 = length (filter (==Player 2) ws)
      in if p1 > p2
            then Player 1
            else if p2 > p1
                    then Player 2
                    else Draw
```

12. (a) Möjlig lösning:

```
intersect([],_,[]).
intersect([X|XS],YS,[X|ZS]) :-
    member(X,YS),
    \+ (member(X,XS)),
    intersect(XS,YS,ZS),
    !.
intersect([_|XS],YS,ZS) :-
    intersect(XS,YS,ZS).
```

(b) Möjlig lösning:

```
common_max(XS,YS,M) :-
    intersect(XS,YS,ZS),
    max_list(ZS,M).
```