Facit och kommentarer till tentamen 2023-04-24 i DA4003

Del 1: flervalsfrågor (1p per fråga)

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

Del 2: kodfrågor

9. (a) Möjlig lösning:

```
void matrix_encrypt_while(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;
  while (i < cols) {
    int j = 0;
    while (j < rows) {
      // 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]);

      j++;
    }
    i++;
  }

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

(b) Möjlig lösning:

```
void matrix_encrypt_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;
 outer:
  if (i < cols) {
    int j = 0;
  inner:
    if (j < rows) {
```

```
      // 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]);

      j++;
      goto inner;
    }
    i++;
    goto outer;
  }

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

10. Möjlig lösning:

```
class MatrixEncrypter {

    private int rows;
    private int cols;

    public MatrixEncrypter(int r, int c) {
        rows = r;
        cols = c;
    }

    public void encrypt(String s) {
        int l = s.length();

        if (l > rows * cols) {
            System.out.println("Error: the string is too long for the matrix");
        } else {
            String out = "";
            for(int i = 0; i < cols; i++) {
                for (int j = 0; j < rows; j++) {
                    // calculate the index to fetch
                    int k = j * cols + i;
                    if (k < l)
                        out += s.charAt(k);
                }
            }
            System.out.println(out);
        }
    }
}
```

11. (a) Möjlig lösning:

```
partition :: (a -> Bool) -> [a] -> ([a],[a])
partition p xs = (filter p xs,filter (not . p) xs)
```

(b) Möjlig lösning:

```
data Files = File String | Folder String [Files]

find :: String -> Files -> Bool
find x (File f) = x == f
find x (Folder _ fs) = any (find x) fs
```

```
folders :: Files -> [String]
folders (File _) = []
folders (Folder x fs) = x : concatMap folders fs
```

12. (a) Möjlig lösning:

```
insert(X,[],[X]).
insert(X,[Y|YS],[X,Y|YS]) :- X =< Y.
insert(X,[Y|YS],[Y|ZS]) :- X > Y, insert(X,YS,ZS).
```

(b) Möjlig lösning:

```
sort_list([],[]).
sort_list([X|XS],YS) :-
    sort_list(XS,ZS),
    insert(X,ZS,YS).
```