# Facit och kommentarer till prov 2022-01-14 i DA2004

## Del A: flervalsfrågor

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

## Del B: kodfrågor

11. Rövarspråksfunktionen ser ut så här:

```
def sve2rovar(s):
    r = ''
    for c in s:
        if c in 'aeiouyåäö':
            r += c
        else:
            r += c + 'o' + c
    return r
```

12. Problemet med `hemlig_kod` är att modulo-operatorn `%` tillämpas på en sträng i **`if`**-satsen.

    Man kan rätta till felet genom att antingen använda typkonvertering, **`int`**`(a[i])`, för att tillämpa modulo på siffrorna i strängen, eller använda funktionen **`ord`** (med **`ord`**`(a[i])`) för att plocka fram ASCII-koden för siffrorna i strängen. Udda siffror råkar ha udda kod, så modulo 2 blir resultatet detsamma.

    Koden ni kan ha sett i tunnelbanan liknar C och Java, men så vitt jag förstår är det inte ett verkligt programmeringsspråk.

13. Möjlig lösning:

```
def div_by_five(l):
    l_div = []
    for n in l:
        if n % 5 == 0:
            l_div.append(n)
    return l_div
```

14. En exempellösning:

```
def tabulate(table):
    for key, val in table.items():
        print(key, val)
```

**15**. En exempellösning:

```
def count_true_assignments():
    n_true = 0
    for x in [False, True]:
        for y in [False, True]:
            for z in [False, True]:
                if x and (y or z) and (not y or not z):
                    n_true += 1
    return n_true
```

**16**. Den här typen av analys görs ibland för att identifiera förvanskad eller påhittad forskning. I de fall jag har läst om har man dock nöjt sig med att göra det med vanlig inspektion. Det finns en del system utvecklade för att automatiskt identifiera forskningsfusk, och det här är en sorts analys man skulle kunna tänka sig i ett sådant system.

Exempellösning:

```
def repeated_numbers(lst):
    registry = dict()
    for num in lst:
        if num in registry:
            registry[num] += 1
        else:
            registry[num] = 1

    duplicated = dict()
    for num, count in registry.items():
        if count > 1:
            duplicated[num] = count
    return duplicated
```

**17**. Möjlig lösning:

```
def sum_rec(s):
    if len(s) == 1:
        return s.pop()
    else:
        return s[-1]+ sum_rec(s[:-1])
```

**18**. Möjlig lösning:

```
def f(lis):
    return sum(map(lambda x: x**2, lis))
```

**19**. Exempel

```
class Card:
    def __init__(self, color, rank):
        if 1 <= color <= 4:
            self.color = color
        else:
            raise ValueError
        if 1 <= rank <= 13:
            self.rank = rank
        else:
            raise ValueError
```

**20**. Till exempel

```
class Card:
    def __init__(self, color, rank):
        if 1 <= color <= 4:
            self.color = color
        else:
            raise ValueError
        if 1 <= rank <= 13:
            self.rank = rank
        else:
            raise ValueError

    def __lt__(self, other):
        if self.rank == 1:
            r1 = 14
        else:
            r1 = self.rank

        if other.rank == 1:
            r2 = 14
        else:
            r2 = other.rank

        c1 , c2 = self.color , other.color

        return (r1 < r2) or (r1 == r2 and c1 < c2)
```