Øving 0.5

Jeg har egentlig ingen anelse om hvordan øvingsinnleveringer er best formaterte, så her kommer ..dette. Tar *veldig gjerne* kritikk på formatering.

1 Kodeforståelse / Oversett til Python

1.a isFibonacciNumber

```
1  def isFibonacciNumber(n):
2      a, b = 0, 1
3      while b < n:
4          temp = b
5          b += a
6          a = temp
7      if b is n:
8          return True
9      else:
10         return False
11
12 # må nesten teste koden
13 if __name__ == "__main__":
14     for e in [21, 23, 18, 3, 5, 4]:
15         if isFibonacciNumber(e) == True:
16             print("1")
17         else:
18             print("0")
```

2 Oversett fra Python til C++

La inn main() for de funksjonene hvor det virket hensiktsmessig å unit-teste litt grovt.

2.a Fibonaccirekker

```cpp
#include<iostream>
using namespace std;

/*
def fibonacci(n):
    a, b = 0, 1
    print("Fibonacci numbers:")
    for x in range(1,n):
        temp = b
        b = a + b
        a = temp
        print(x,b)
    print()
    return b
*/

int fibonacci(int n) {
    int a = 0, b = 1, temp = 0;
    cout << "Fibonacci numbers:" << endl;
    for (int i = 1; i < n; i++) {
        temp = b;
        b = a + b;
        a = temp;
        cout << i << " " << b << endl;
        }
    return b;
}

//Må jo nesten kjøre, sånn bare for å teste.
int main() {
    fibonacci(8);
    return 0;
}
```

Enda litt morsom oppførsel, men oppførsel som man egentlig forventer. For et gitt input **n**, så lister den opp til n-1, siden range(1, n) forstår sitt input som et halv-åpent intervall fra (1,n].

(Siden halvåpne internvall er definert i stilen av $\forall e \in (1, n], e \geq 1 \wedge e < n$. Så range(1,n) gir [1, 2, 3,... n-1]. Nullindeksering. Hurra!)

2.b Trekanttall

```
1  #include<iostream>
2  using namespace std;
3
4  // Fig. 12
5  void triangleNumbersBelow(int n) {
6      int acc = 1, num = 2;
7      cout << "Triangle numbers below " << n << ":" << endl;
8      while((acc+num) < n){
9          acc = acc+num;
10         num++;
11         cout << acc << endl;
12     }
13 }
14
15 //Fig. 13
16 bool isTriangleNumber(int number){
17     int acc = 1;
18     while(number > 0) {
19         number = number - acc;
20         acc = acc + 1;
21     }
22     if(number == 0) {
23         return true;
24     } else {
25         return false;
26     }
27 }
```

Pythonkoden supplert har ikke et return-uttrykk for triangleNumbersBelow(), så jeg gikk for void().

2.c Sum av kvadrerte tall

```
1  #include<iostream>
2  using namespace std;
3
4  int squareNumberSum(int n) {
5      int totalSum = 0;
6      int e = 0;
7      for(int i = 0; i < n; i++) {
8          totalSum += i*i;
9          e = i+1;
10         cout << (e*e) << endl;
11     }
12     cout << totalSum << endl;
13     return totalSum;
14 }
15
16 int main() {
17     squareNumberSum(6);
18     return 0;
19 }
```

Jeg mener at pythonkoden har en litt besynderlig oppførsel, i at den printer ut kvadratet av *neste* **i**, for hver iterasjon. Slik at, for et input **6**, så gir den totalSum som 55, men produserer en liste som går mot 91. Se f.eks.

```
dhcp-037213:kode mathiasholm$ py test.py
1
4
9
16
25
36
55
```

2.d Størst av to tall

```
1  #include<iostream>
2  using namespace std;
3
4  int max(int a, int b) {
5      if(a > b) {
6          cout << "a is greater than b" << endl;
7          return a;
8      } else {
9          cout << "a is greater than, or equal, to b" << endl;
10         return b;
11     }
12 }
13
14 int main() {
15     max(2, 4);
16     max(2, 2);
17     max(4, 2);
18     max(6, 8);
19     return 0;
20 }
```

Hastig unit-testing på i main()

2.e Primtall 1 & 2

```
1  #include<iostream>
2  using namespace std;
3
4  //2e - Primtall 1
5  bool isPrime(int n) {
6      bool primeness = true;
7      for(int i = 2; i < n; i++) {
8          if(n%i == 0) {
9              primeness = false;
10             return primeness;
11         }
12     }
13     return primeness;
14 }
15
16 //2f - Primtall 2
17 void naivePrimeNumberSearch(int n) {
18     for(int i = 2; i < n; i++) {
19         if(isPrime(i)) {
20             cout << i << " is a prime" << endl;
21         }
22     }
23 }
24
25 int main() {
26     int derp[8] = {2, 3, 4, 6, 7, 42, 41, 40};
27     for(int i = 0; i < 8; i++) {isPrime(derp[i]);}
28     for(int i = 0; i < 8; i++) {naivePrimeNumberSearch(derp[i]);}
29     return 0;
30 }
```

Størrelsen på unit-testene ble litt keitete.
Klistret 2e og 2f inn på samme side, siden 2f har et kall til isPrime()

2.g Største Fellesnevner

```
1  #include<iostream>
2  using namespace std;
3
4  int findGreatestDivisor(int n) {
5      if(n < 0) {  n = n*-1; }
6      for(int i = n-1; n > 0; i--) {
7          if(n%i == 0) {
8              return i;
9          }
10     }
11     return 0;
12 }
13
14 int main() {
15     int test[8] = {-2, 3, 4, 6, 7, 42, 41, 40};
16     for(int i = 0; i < 8; i++) {
17         cout << findGreatestDivisor(test[i]) << endl;
18     }
19     return 0;
20 }
```

Introduserte linje 5 for å gi støtte for negativt input. Hvori linje 10 helst aldri skal nås i normal programflyt, men som er *nødvendigish* for å unngå en kompileradvarsel.

2.h Telling med lister

```
#include<iostream>
using namespace std;

void compareListOfNumbers(int len, int liste[]) {
    int r[3] = {0, 0, 0};
    for(int i = 0; i < len; i++) {
        if(liste[i] < 0) {
            r[0]++;
        }
        else if(liste[i] == 0) {
            r[1]++;
        }
        else {
            r[2]++;
        }
    }
    cout << r[0] << " numbers were below zero" << endl;
    cout << r[1] << " numbers were zero" << endl;
    cout << r[2] << " numbers were above zero" << endl;
}


int main() {
    int test[5] = {1, 2, 3, 0, -4};
    int len = sizeof(test)/sizeof(*test);
    compareListOfNumbers(len, test);
    return 0;
}
```

Så, siden C++ er statisk, i motsetning til pythons herlige dynamiske verden, så kan ikke oppførselen med en liste av arbitrær lengde overføres like enkelt fra pythonkoden til C++. Det er veldig tidlig i semesteret, og vi har ikke vært borti pekere enda, og vektorer virket litt overkill å begynne med, så jeg gambler på få aksept for at jeg introduserte et implisitt ekstraparameter for listen, så svaret mitt skulle reprodusere oppførselen til pythonkoden så nært som mulig, uten å knote for mye.