Programming for computerteknologi
Hand-in Assignment Exercises

Week 11: Verifying Correctness of Recursive Programs

Exercise 1)
We have been given a recursive factorial function and we now need to prove that the function is correct using proof of induction.

Proof by Induction

Base case:

$fact(1) = 1$

Inductive step:

$fact(n) = n \cdot fact(n-1)$

IF $fact(n-1)$ is correct THEN $fact(n)$ is correct as well.

To prove the case for n is correct, we need to prove for cases smaller than n.

IF $fact(4)$ is correct THEN $fact(5)$ is correct.

IF $fact(3)$ is correct THEN $fact(4)$ is correct.

IF $fact(2)$ is correct THEN $fact(3)$ is correct.

IF $fact(1)$ is correct THEN $fact(2)$ is correct.

Since we have already proved $fact(1)$ to be correct the rest of the cases declared above must be correct as well all the way up to $fact(n)$. Therefore $fact(n)$ is correct for any positive integer n we give the function as input.

Exercise 2)
We have been given a proof that the sum of the first positive n odd numbers is equal to n^2. With inspiration from the proof, we now have to make a recursive function that calculates the sum of the first positive n odd numbers. The function will look like the following:

```
int sumn (int n)
{
    /* pre-condition */
    assert(n > 0);

    /* post-condition */
    if(n == 1) { // base case
        return 1;
    } else { // inductive step
        return 2 * n - 1 + sumn(n - 1);
    }
}
```

Exercise 3)

We have to convert a recursive function that sums the integers 1 to n into (a) an equivalent tail recursive program, and (b) a program using a while loop. First, we have the tail recursive program:

```
/* Sum integers 1 to n */
int sumtail (int n, int total)
{
  /* pre-condition */
  assert(n > 0);

  /* post-condition */
  if(n > 1) {
    return sumtail(n - 1, total + n);
  } else {
    return ++total;
  }
}
```

Next up we have the program using a while loop to sum integers 1 to n.

```
/* Sum integers 1 to n */
int sumwhile (int n)
{
  int r = 0;
  while(n > 0) {
    r += n;
    n--;
  }
  return r;
}
```

To check if the functions run correctly, I have made a couple of test cases.

```
/* Exercise 3 */
TEST_CASE("sumtail")
{
    REQUIRE(sumtail(1,0)==1); // sumwhile(1) must be 1
    REQUIRE(sumtail(7,0)==28); // sumwhile(33) =1+2+3+4+5+6+7 = 28
    REQUIRE(sumtail(33,0)==561); // sumwhile(33) =1+2+3+...+33 = 561
}

/* Exercise 3 */
TEST_CASE("sumwhile")
{
    REQUIRE(sumtail(1,0)==1); // sumwhile(1) must be 1
    REQUIRE(sumtail(7,0)==28); // sumwhile(7) = 1+2+3+4+5+6+7 = 28
    REQUIRE(sumtail(33,0)==561); // sumwhile(33) =1+2+3+...+33 = 561
}
```

I have chosen these test cases to check if the base case works, if the function works with smaller numbers and if it works with greater numbers.

Exercise 4)

We have been given the task to convert a recursive Fibonacci function into an equivalent tail recursive program.

```
int fib (int n, int p, int pp)
{
    /* pre-condition */
    assert(n > 0);

    /* post-condition */
    if(n < 3) { // base case -> returns the final Fibonacci number
        return pp + p;
    } else { // inductive step
        return fib(n - 1, pp, p + pp);
    }
}
```


AARHUS UNIVERSITET
AU ENGINEERING

I have made a couple of test cases to test if the program runs correctly.

```
/* Exercise 4 */
TEST_CASE("fib")
{
    REQUIRE(fib(1,0,1)==1); // fib(1) must be 1 (base case)
    REQUIRE(fib(2,0,1)==1); // fib(2) must be 1 as well (base case)
    REQUIRE(fib(7,0,1)==13);
    REQUIRE(fib(15,0,1)==610);
}
```

The first two test cases tests if the base cases run correctly. Then I test for a smaller number and for a greater number to see if the cases runs correctly as well.