AARHUS UNIVERSITY JONAS ASKJÆR-JØRGENSEN DEADLINE: 31/10
AU-id: AU713502 Studienr: 202207230

Programmering for Computerteknologi

Hand-in Assignment Exercises

Week 8: Designing Sequences of Program Instructions for Solving Problems

1) (Text answer) Consider the following program for computing Factorial numbers:

```
/* Factorial function definition */
int fact(int n)
{
  int i; /* counter variable */
  int f; /* factorial */

  /* pre-condition */
  assert (n >= 0);

  /* post-condition */
  f = 1;
  for(i = 1; i <= n; i = i + 1)
  {
    f = i * f;
  }
  return f;
}
```

Provide your answers to the following questions in a plain text file:

a) How many arithmetic operations are required to compute $fact(5)$?

First off it is required to add 1 to i five times because i is used our count for how many times we have computed the factorial. Then furthermore we have $i \cdot f$. In the end it should require ten arithmetic operations to compute factorial(5).

b) How many arithmetic operations are required to compute $fact(n)$ for any positive integer n?

To setup a sentence that would be accurate for any iteration of $fact(n)$.
It should be $fact(n) = n + n$, because it calculates for $i++$ and $f = i \cdot f$. Both are equal to n.

2) (Code answer) Implement an insertion sort function that is used for a singly linked list of integers, so that the final list is from smallest to largest.

See insertion_sort.c to see how I have implemented my sort function for a singly linked list of integers. The list computes correctly the smallest to the largest and the tests pass correctly.

3) (Code answer) Queues maintain, (FIFO).

a) Implement a queue based on singly linked lists as discussed in the lecture. That is, implement the four functions mentioned above.

See my code in list_queue.c, I have implemented all four functions.

b) Write tests to verify your implementation as presented in the lectures (Please review Lecture 3, Section "Testability of a program" and "Testing functions with functions"). Your tests should ensure the following "Laws" hold for any implementation of a queue.

The tests are already written, but my four functions work with the tests and all tests pass.

(A) After executing `init_queue(q);` the queue `s` must be empty.
(B) After executing `enqueue(q,x); y = dequeue(q);` the queue `q` must be the same as before execution of the two commands, and `x` must equal `y`.
(C) After executing
```
enqueue(q, x0); enqueue(q, x1);
y0 = dequeue(q); y1 = dequeue(q);
```
the queue `q` must be the same as before execution of the two commands, `x0` must equal `y0`, and `x1` must equal `y1`.