

# Øving3

## TDT4200

Anette Fossum Morken

### Problem 1, Debugging

**a**

Koden frigjør ikke igjen minnet når programmet er ferdig med å kjøre.

**b**

**c**

- Linje 10: Allokerer minnet for usignet char når det som skal være det er char.
- Linje 13: Gir ikke lastChar noen verdi.
- Linje 17: for-løkken går fra 10 til 0 noe som gjør at den looper gjennom 11 elementer.

Andre feil ved koden:

- Når pekeren mem er opprettet peker den på et sted i minnet, men verdiene til dette området er ukjent, slik at når man setter inn i stringen og om den er kortere enn 10 elementer vil det som var der etter lengden til stringen fortsatt være der. (satte inn `memset(mem, sizeof(char), 10);` etter `malloc`).
- Programmet gir sjekker ikke og gir ikke ut noen feilmelding om stringen er lengre enn 10 karakterer.
- Når inputet er lengre en 10 elementer kopierer programmet de 10 første elementene i inputet og printer ut disse ti i motesatt rekkefølge.

### Problem 2, optimalisering

**d**

Kjøretid på Vilje:

**e**

Bildene er ikke like i følge `checker.c`, men det er innenfor feilmarginen. Feilen skyldes at for å øke antall elementer på L1 er det redusert fra dobbel presisjon til singel presisjon.

## f

`sum+=imageIn->data[...]` gjøres 11512800 ganger for en farge når *size* = 2 og 39081600 ganger når *size* = 8.

## g

Endringer som er gjort:

- Når `performNewIdeaIteration` kjøres gjør den utregninger for alle tre fargene, ikke for en og en slik den var før.
- for-løkkene er delt opp slik at randbetingelsene behandles separat.
- programmet legger først sammen alle pikslene *size* fra en piksel i x-retning for alle pikslene og lagrer denne i et midlertidig bilde. Deretter går den gjennom dette midlertidige bildet og summerer alle pikslene i y-retning i det midlertidige bildet for så å lagre det i en nytt midlertidig bilde. samtidig som denne summen lagres i det nye midlertidige bildet finnes gjennomsnittet av summen til pikselen og lagres i det samme bildet som ble sendt inn.
- alle summeringer blir gjort ved at summen i pikselene når `senterX=0` (når det summeres i x-retning) og `senterY=0` (når det summeres i y-retning) blir regnet ut ved bruk av for-løkke. Resten blir regnet ut ved å legge til pikselen *size* bortenfor (til høyre eller under pikselen det regnes ut for) og rekke fra pikselen *size*+1 før (til venstre eller over pikselen det regnes ut for).
- Når det regnes ut i y retning gjøres dette også radvis bortover.
- Det er lagt til en kode `copyImage` som kopierer alle verdiene fra et allerede konvertert bilde. Dette gjør at man slipper å konvertere bildet fra ppm til flyttall mer enn en gang.
- Det brukes single presisjon i stedet for dobbel.

Det er også en optimalisering som jeg ikke har gjort, det er å gjøre endringer på `performNewIdeaFinalization`, denne kan gjøres mer effektiv, men det er utenfor min kompetanse C.