Oppgave 3 Dokumentasjon

I denne oppgaven var jeg heldig å hadde gjort mye av grunnarbeidet på forhånd, så det jeg startet med var å jobbe ut ifra den koden jeg allerede hadde. Lagde noen nye filer og gjorde om litt på makefilen selvfølgelig.

Det var noe av kodesnutten min som ikke fungerte optimalt fra før av, så gjorde litt forbedringer der til å starte med.
I den oppgaven jeg hadde gjort tidligere så var input hardkodet i main filen (ikke fra bruker input) så til å starte med så prøvde jeg meg frem med hardkodet input, og startet å lage de nye funksjonene som eksamensoppgaven krevde. Da brukte jeg forsovet også den gamle structen som var litt annerledes enn hva oppgaven sa.

Jeg startet med å lage funksjonen for å "Legge til et element i listen" noe som var ganske enkelt.
så lenge man har en struct med pNext,pPrev, og en som holdt styr på hva som var head og tail. Så var det bare å flytte pekerne til det riktige elementet.
(om det ikke var noe fra før av, så ble head og tail = nyNode) og om man skulle legge til noe etter der så var det bare å sette pekeren fra forrige node til nyNode og sette nyNode som en tail, osv..

"Hente ut element N fra listen" hvor første element var "index 1", så var det bare å loope igjennom listen (til den ble null) med en int som holdt styr på hvilken index man var på (som startet på 1 selvfølgelig) og printet bare ut all informasjon om det elementet når man traff den indexen

"Finne element som matcher navn i listen", mye av det samme som det over, bare at man leter etter node->name.

"Slette alle elementer i listen som har et gitt navn" litt av det samme.
- loope igjennom listen
- sjekke om node->name = inputNavn
- flytte litt på pekere (node->pPrev->pNext = node->pNext) og motsatt, for å "fjerne" noden
- free()

"Sletter alle elementer som har en alder lavere(som jeg valgte) enn input", mye av det samme her også
- loope igjennom listen
- sjekke om (node->age < N) og fjerne noden på samme måte som over.

og hvis man fant noden så huske å bruke free()

Etter jeg hadde laget disse funksjonene så begynte jeg å fikse på structen sånn at det ble riktig (var vel nesten bare å bytte litt navn + legge til en til variabel)
Så fant jeg frem en av switch-case oppgavene jeg hadde gjort tidligere og fjernet den hardkodede "input" jeg hadde og spurte heller bruker om de kunne taste inn et tall mellom 1-6, med forksjellige alternativer for hvert tall.

og så flyttet jeg inn de riktige funksjonene til å gjøre hva de skulle om bruker tastet inn "riktig" tall.

Siden jeg brukte getchar() så valgte jeg å escape unna '\n' ved å sjekke om det i øverst av while loopen, for så å "skippe" switch-casen med en case for "\n" som bare breaker.

Jeg valgte å spørre bruker om de vil fortsette for ryddighetens skyld (hva jeg syntes passet best ihvertfall), while-loopen min vil avslutte om input = 6, eller om da bruker svarer noe annent enn “ja” når det spørres om bruker vil fortsette, hvis bruker ikke da svarer ja, så setter jeg bare input = 6, sånn at while loopen breaker, og jeg kommer ut av loopen, sånn at jeg får bladd igjennom linked listen og får frigjort minne til hver node.

Jeg har også valgt å ha 2 .c filer, en som inneholder main() og printf-delen og en fil som inneholdt funksjonene, siden det ble så mange funksjoner. Begge har jo da #include “Oppgave3.h”

Helt til slutt fikset jeg litt opp i noen bugs som oppsto etter bruker hadde gjort litt forskjellige alternativer :)

Vil få noen warnings når man compiler koden, fordi jeg ikke "sjekker" return valuen til scanf (Kunne "fikset" de ved å lage noen if-setninger til scanf(), men det ble uoversiktlig og unødvendig) derfor har det ikke noe å si i dette tilfelle.(mulig din kompiler ikke klager på dette)