Oppgave 1

I C-koden opprettes det en FIFO (First-in-first-out) som er en metode for å håndtere rekkefølger på. Dette skjer ved at det opprettes et minnebuffer som lagrer meldinger “message”. Etter at det i *main* opprettes en barneprosess med *fork* sjekkes det deretter om barneprosessen kjøres (pid == 0), og i så fall settes skriveprosessen (write-funksjonen) i gang. Ved hjelp av FIFO sørger dette for at foreldreprosessen venter på at barneprosessen fullfører (wait(NULL)). Til slutt slettes det avsatte minnebufferet når programmet har kjørt ferdig.

Oppgave 3

a

```
[halambique@bluefish lab_07]$ ./read_shm ; ./write_shm
sum1 = 0, sum2 = 50000005000000
```

Siden de kjøres sekvensielt, men med read-programmet først, har ikke sum1 fått en verdi i det minnet programmene deler med shm_objektet.

b

```
[halambique@bluefish lab_07]$ ./write_shm ; ./read_shm
sum1 = 50000005000000, sum2 = 50000005000000
```

Her kjøres programmene ved å skrive først, og selv om det er ulike måter å nå summen på, blir det like resultater.

Sum1 lages ved at A[] mappes til det delte minneobjektet, og dette blir da et array med verdier fra 0-N. I read_shm summeres disse integerne. Sum2 er ikke avhengig av å lese A[] (fra write_shm), og gjør derfor sin operasjon uavhengig av rekkefølgen.

c

```
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 38448212221676, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 39333893865318, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 37545336745253, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 37864229779478, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 37965471459769, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 37949620854182, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 37911675891904, sum2 = 50000005000000
[halambique@bluefish lab_07]$ (./write_shm &) ; ./read_shm
sum1 = 38245144292182, sum2 = 50000005000000
```

Her får vi noenlunde like resultater på sum1 hver gang programmene kjøres. Dette forteller oss noe om at når vi kjører dem synkronisert, vil ikke write-programmet rekke å fylle arrayet A[] før read-programmet skriver ut resultatene.