Del 2: Teori

Uppgifter

1. **Notera:** För G på tentan behöver du få godkänt på denna uppgift.

   Tillsammans med det här dokumentet har du fått två lösningar på problemet i del 1 i filerna `a.c` och `b.c`. Dessa lösningar har olika styrkor och svagheter, och har i vissa fall inte ens lyckats synkronisera programmet korrekt.

   Välj en av de ovanstående lösningarna och jämför den med din lösning av del 1. Innan du börjar, fundera på vilken lösning som är mest lämplig att jämföra med. För att kunna svara bra på den här frågan vill du välja en lösning som är tillräckligt olik din lösning, annars kommer du inte kunna svara tillfredsställande på frågorna nedan. Om du inser att din lösning på del 1 innehåller fel, välj då en lösning som löser så många av felen i din lösning som möjligt och beskriv varför din lösning är fel och den givna lösningen är korrekt och vice versa.

   Jämför nedastående egenskaper i din lösning och i lösningen du har valt. Skriv sedan ner vad du kom fram till och hur du resonerade för att komma fram till ditt svar, gärna uppdelat i stycken eller rubriker som motsvarar punkterna nedan.

   (a) Innehåller antingen din lösning eller den givna lösningen du har valt några synkroniseringsproblem? Beskriv (gärna med exempel) vad som kan gå fel i den ena, och varför det inte kan hända i den andra.

   (b) Innehåller antingen din lösning eller den givna lösningen du har valt några *busy-wait*? Beskriv (gärna med exempel) hur de kan uppkomma, eller hur lösningen har sett till att den *busy-wait* som fanns i den givna koden inte kan inträffa.

   (c) Använd de fyra villkoren för deadlock för att komma fram till om det finns risk för deadlock i din lösning, och i den givna lösningen du har valt.

   (d) Fundera på de olika lösningarnas prestanda. Vilken av de tre lösningarna (din lösning, `a.c` och `b.c`) förväntar du dig blir klar med sitt arbete snabbast? Varför? Kan du förbättra din lösning så att den blir snabbare?

   Tips: Tänk på att det huvudsakligen är `task_create` som tar tid.

   För godkänt på den här uppgiften vill vi se:

   - Att ditt svar är mellan 700 och 1500 ord, exklusive eventuella kodexempel.
   - Att du har diskuterat alla punkter som finns i uppgiften ovan.
   - Att du kan förklara varför din lösning är korrekt, eller med hjälp av en referenslösning kan förklara varför din lösning är felaktig och hur den kan korrigeras.
   - Att du kan förklara skillnaden mellan din lösning och den valda givna lösningen med avseende på korrekthet och möjlig parallellism.

2. När programmet arbetar skriver det ut statusmeddelanden för att visa hur långt det har kommit. Vi förväntar oss att se: [4p]

```
1 av 12 klara...
2 av 12 klara...
3 av 12 klara...
...
11 av 12 klara...
12 av 12 klara...
```

   Är det **garanterat** att vi alltid ser 12 rader med alla tal mellan 1 och 12 i både `a.c` och `b.c`? Motivera för båda lösningarna antingen varför vi vet att detta alltid är fallet, eller beskriv (gärna med exempel) vad som kan hända i stället.

3. I ett system körs tre processer, *P1, P2* och *P3*. I systemet finns det också tre typer av resurser, *A, B* och *C*. Tabell 1 visar hur många resurser som finns, hur systemets resurser är allokerade i nuläget och det maximala resursbehovet för varje process. [2p]

	A	B	C
P1	10	7	11
P2	7	2	1
P3	7	7	6

Maximal resursanvändning

	A	B	C
P1	7	5	8
P2	6	0	0
P3	4	4	3

Nuvarande resursanvändning

	A	B	C
P1	19	11	14

Totalt antal resurser

Tabell 1: Resurser i systemet.

   Process *P3* begär en extra resurs av typ *C*. Ska begäran tillåtas enligt Banker's algorithm? Redovisa dina beräkningar.

4. Lös den här uppgiften genom att modifiera en kopia av filen `b.c`, och skicka in den separat i Lisam.

   (a) Ta bort arrayen av lås (`tasks_lock`) i din kopia av `b.c` och ersätt synkroniseringen med lämpliga atomiska operationer från filen `wrap/atomics.h`. [2p]

   (b) Ta bort semaforen (`num_generated`) och ersätt den med en heltalsvariabel (en `int`) och lämpliga atomiska operationer från filen `wrap/atomics.h` för att manipulera den. Din lösning får innehålla *busy-wait*, men ska inte skriva ut dubletter av raden `X av 12 klara...` [2p]

   Notera: Det går att använda `atomic_swap` och `compare_and_swap` på pekarvariabler, trots att exempelimplementationerna i `wrap/atomics.h` endast hanterar heltal.