Instuderingsfrågor

1. Skriv ett program i Go som skapar en goroutine som skriver ut en text på skärmen
2. Skriv ett program i Go som läser ett värde från en kanal och skriver ut det på skärmen
3. Var är *GIL* i Python och hur påverkar det trådade program i Python?
4. Skriv ett program i Go som kan ge upphov till ett *data race*
5. Implementera *Peterson's* algorithm i Go-liknande kod.
6. Implementera en producent i Go-liknande kod som använder mutex för ömesidig uteslutning
7. Implementera en producent i Go-liknande kod som använder kanaler för ömesidig uteslutning
8. Ge ett exempel i Go där två goroutines delar en variabel och använder en mutex för att (korrekt) skydda tillgången
9. Ge ett exempel på ett program i Go med kanaler som kan leda till *deadlock*
10. Ge ett exempel på ett program i Go utan kanaler som kan leda till *deadlock*
11. Implementera en parallel *prefix sum* med goroutines i Go
12. Implementera en parallel *prefix scan* med goroutines i Go
13. Implementera en parallel *odd-even transposition sort* med goroutines i Go
14. Visa hur en *for*-loop kan paralleliseras i Go med goroutines
15. Hur kan en *barrier* impementeras med mutex? Visa i Go-liknande kod
16. Hur kan en *barrier* impementeras med kanaler? Visa i Go-liknande kod
17. Ge en concurrent/parallel algoritm för att hitta den minsta värdet i en lista i Go-liknande kod
18. Vad innebär en atomär variabel i Go
19. Visa hur en variabels värde kan jämföras och bytas atomärt i Go
20. Förklara vad *happens before* i en minnesmodell
21. Vilka antaganden kan göras om två trådar modiferar en delad variabel (enligt Javas minnesmodell).
22. Förklara begreppet *multicore cpu*
23. Vad är skillnanden mellan *concurrency* och *parallelism*?
24. Vad betyder det att ett problem är *embarrassingly parallel*?
25. Ge exempel på ett problem som är *embarrassingly parallel*
26. Förklara begreppet *operativsystem*
27. Vad betyder det att ett operativsystem kan ses som ett interface mot datorn?
28. Vad betyder det att ett operativsystem kan ses som ett kontrollsystem för datorn?
29. Vad är skillnaden på *system mode* och *user mode*?
30. Ange fem tillstånd en process kan befinna sig i och förklara vad de innebär
31. Förklara begreppen *stack*, *heap*, *data* och *text* i relation till processer
32. Vad lagras i ett processkontrollblock (PCB)?

33. Vad är en *context switch*?
34. Förklara vad som händer under en *context switch*
35. Vad innebär schemaläggning av en process?
36. När tas beslut om schemaläggning?
37. Vad gör *dispatcher*?
38. Ange tre kriterier som kan användas för att bestämma vilken schemaläggningsalgoritm som skall användas
39. Vad betyder det att en process är *I/O bound*?
40. Vad betyder det att en process är *CPU bound*?
41. Förklara hur schemaläggning sker om *First Come, First Served* används
42. Förklara hur schemaläggning sker om *Round Robin* används
43. Förklara *preemptive* schemaläggning
44. Förklara *non-preemptive* schemaläggning
45. Vad är ett *quantum*?
46. Hur påverkar ett *quantum* schemaläggningen?
47. Vad är viktigt att tänka på när man bestämmer *quantum*?
48. Förklara begreppen *isolation* och *encapsulation* i relation till processer
49. Vad är skillnaden på en process och en tråd?
50. Vad används stacken till i en tråd?
51. Var är skillnaden på *user-level-* och *kernel-level*-trådar?
52. Förklara begreppet blockerande anrop
53. Vad händer om en *user-level* tråd gör ett blockerande anrop?
54. Vad händer om en *kernel-level* tråd gör ett blockerande anrop?
55. Varför vill man dela minne mellan trådar?
56. Vad är skillnaden mellan att dela minne mellan trådar och processer?
57. Förklara begreppet *sequentially consistent* i relation till minne
58. Vad innebär ett *data race/race condition*?
59. Vad är en kritisk sektion?
60. Ge exempel på ett *data race*
61. Vad innebär ömesidig uteslutning?
62. Hur kan ömesidig uteslutning lösas? ge exempel.
63. Förklara hur *strict alteration* försöker lösa ömesidig uteslutning. Vilka problem finns?
64. Förklara *Peterson's* algoritm för ömsesidig uteslutning
65. Vad innebär progression i relation till ömesidig uteslutning?
66. Vad innebär begränsad väntan i relation till ömesidig uteslutning?
67. Vad är en semafor?
68. Vad är ett *mutex lock*?
69. Vad är skillnaden mellan en binär och en räknande semafor?
70. Visa hur en räknande semafor kan implementeras med hjälp av binära semaforer
71. Vilka risker finns det med låsning?
72. Vad är ett *deadlock*?
73. Vad innebär det att en process är *deadlocked*?
74. Vilka fyra villkor krävs för *deadlock*?
75. Förklara *hold and wait*
76. Förklara *no preemption*
77. Förklara *circular wait*

78. Förklara hur *deadlock* kan förhindras
79. Hur kan *mutual exclusion* förhindras för att undvika *deadlock*?
80. Hur kan *hold and wait* förhindras för att undvika *deadlock*?
81. Hur kan *no preemption* förhindras för att undvika *deadlock*?
82. Hur kan *circular wait* förhindras för att undvika *deadlock*?
83. Ge exempel på hur en semafor kan användas så att *hold and wait* inte gäller
84. Varför kan det vara problematiskt att förhindra *no preemption*
85. Varför kan det vara problematiskt att förhindra *hold and wait*
86. Förklara *starvation*. Varför är det ett problem?
87. Vad är en *resource allocation graph* och hur används den i samband med *deadlock*?
88. Hur kan man avgöra om ett system är i *deadlock* med hjälp av en *resource allocation graph*?
89. Är det ok att ignorera *deadlock*?
90. Vad är skillnaden på att förhindra och undvika *deadlock*?
91. Förklara *Banker's* algoritm
92. Hur kan ett system återhämta sig från *deadlock*?
93. Förklara *prefix sum*
94. Förklara *prefix scan*
95. Förklara *odd-even transposition sort*
96. Varför är det en dålig idé att skapa nya trådar för varje rekursivt anrop i en *divide and conquer*-algoritm?
97. Varför kan det vara svårt att parallelisera rekursiva algoritmer?
98. Vad är en *barrier*
99. Hur kan en semafor användas för att signalera mellan trådar?
100. Förklara hur *depth-first search* kan paralleliseras med trådar
101. Förklara hur *breadth-first search* kan paralleliseras med trådar
102. Förklara hur *Prim's* algoritm kan paralleliseras med trådar
103. Vi kan hitta det minsta värdet i en lista på linjär tid ($O(N)$). Hur lång tid tar det att köra med P procecssorer? Motivera.
104. Vad krävs för att vi skall erhålla en *speedup* på P med en algorim som körs på P procecssorer
105. Är det alltid snabbare att parallelisera en algoritm och köra den på så många processorer som möjligt? Motivera.
106. Förklara *N-ary*-sökning.
107. Förklara hur *N-ary*-sökning kan paralleliseras med trådar
108. Förklara *lock free*
109. Förklara *wait free*
110. Vad menas med en optimistisk algoritm (med avseende på ömesidig uteslutning)?
111. Förklara *compare and set*
112. Visa hur *compare and set* kan användas för att implementera en binär semafor
113. Förklara *hand over hand locking*
114. Vad är nackdelarna med att låsa "för mycket"?
115. Vad är nackdelarna med att låsa "för lite"?
116. Vad är fördelarna med en optimistisk algoritm (med avseende på låsning)?
117. Vad är nackdelarna med en optimistisk algoritm (med avseende på låsning)?
118. Förklara hur radering i en länkad lista kan implementeras med en optimistisk algoritm
119. Ange några problem med att skriva flertrådade program
120. Varför kan det vara svårt att sätta samman (compose) flera flertrådade funktioner?

121. Vad är en *Future*?
122. Förklara begreppet *coroutine*
123. Förklara begreppet *cooperative multitasking*
124. Förklara begreppet *asynchronous programming*
125. Förklara begreppet *callback*
126. Vad händer om en blockerade funktion körs av en funktion på *event loop*
127. När bör man använda *asynchronous programming*?
128. Vad är en kanal i Go?
129. Vad gör *select* i Go?
130. Förklara begreppet *done*-kanal
131. Visa (i Go-liknande kod) hur en goroutine kan avslutas med hjälp av en *done*-kanal
132. Implementera en funktion i Go som bestämmer det största värdet som skickas på en kanal
133. Implementera en funktion i Go som delar en kanal i flera kanaler (multiplex). Tänk på att använda goroutiner
134. Vad är ett closure i Go?
135. Förklara WaitGroup i Go