Instud.-frågor inför tentan i kursen Progr. inbyggda system

Svar till alla frågor kan hittas i kursboken.

Frågorna omfattar kursens **båda delar I (C/C++) & II (Python)**.

Alla sidouppgifter hänvisar till kursboken.

DEL I C/C++

Kap 1-2 Introduktion & Progr. miljöer, sid 6-38

1. *Raspberry PI* nämns ofta som ett exempel på ett inbyggd system. Varför blev det oväntat populärt utanför sitt ursprungliga användningsområde?
2. Formulera en allmän definition för inbyggda system. Ange några exempel som uppfyller din definition. Varför är en mobiltelefon inget inbyggt system?
3. Vilka krav måste inbyggda system uppfylla för att kunna anses vara användbara i praktiken?
4. Vilka operativsystem förekommer hos inbyggda system och vilka är mest populära?
5. Vad betyder *Realtid* inom databehandling och vilken roll spelar begreppet för inbyggda system?
6. Redogör för begreppet *programmerbarhet* hos inbyggda system.
7. I vilka komponenter är mjukvaran hos inbyggda system lagrade?
8. Inom vilket intervall rör sig minnesstorleken för inbyggd programvara?
9. Vilka är de typiska hårdvarukomponenterna hos inbyggda system?
10. Nämn tre olika komponenter hos inbyggda system.
11. Vad betyder *firmware* hos inbyggda system? Ge ett exempel på firmware.
12. Vad är skillnaden mellan *firmware* och *software*?
13. I vilka minnesenheter lagras programvara hos inbyggda system?
14. Vad är skillnaden mellan *inbyggda system* och *Internet of Things (IoT)*.
15. Nämn några exempel på inbyggda system hos motorfordon.
16. Redogör med egna ord hur ett inbyggt operativsystem fungerar.
17. När kom upp de första inbyggda systemen och vilka tillämpningar hade de då?
18. Varför är C/C++ och Python lämpliga språk för programmering av inbyggda system?
19. Nämn några programmeringsmiljöer som är lämpliga för utveckling av C/C++ program.
20. Hur långt går standardiseringen av C/C++ bland de olika utvecklingsmiljöer?
21. Nämn några praktiskt relevanta skillnader mellan utvecklingsmiljöerna *Borland C++ Compiler* och *Visual Studio*.
22. Varför är teckentabellen *Unicode* inte standardiserad i full utsträckning hos de olika C/C++ kompilatorerna?

23. På vilket sätt kan man få korrekta tecken i sina programutskrifter om dessa ligger utanför ASCII-standarden?
24. Vad står *GNU* för och i vilket sammanhang förekommer det?
25. Vilka programvaror behöver man installera för att kunna bygga en egen IDE? Vilka steg måste tas efter installationen?
26. Vilken systemvariabel måste man uppdatera i sitt operativsystem om man vill installera en C/C++ kompilator och köra den från kommandofönstret?
27. Vad är **`cmd`**? Nämn några kommandon som kan användas i **`cmd`** och förklara deras betydelse.
28. Hur byter man mapp i **`cmd`**? Ge ett exempel.
29. Vad måste man göra när man vill köra en exekverbar fil i **`cmd`** som inte finns i den aktuella mapp som **`cmd`**-fönstret står i?
30. Vad visar prompten i Windows' kommandotolk?

Kap 3 C/C++ programmering, sid 39-75

31. Vilket bibliotek måste man inkludera i sitt C/C++ program om man vill anropa funktionen **`MessageBox()`**?
32. Genererar funktionen **`MessageBox()`** en textbaserad eller en grafisk miljö?
33. Kan man anropa funktionen **`MessageBox()`** i en Console Application i Visual Studio?
34. Varför kan programmet **`MessageBoxB`** (sid 40) inte kompileras i Visual Studio och vad är lösningen?
35. Hur kan man få korrekta svenska tecken ä, å, ö, Ä, Å, Ö i meddelanderutorna av **`MessageBox()`**?
36. Vad står *LPCWSTR* för och i vilket sammanhang/varför använde vi det?
37. Vad gör **`L`** i följande kod: **`L"Detta är en MessageBox från C++."`**
38. Är de svenska specialtecknens koder i C/C++ Unicode-koder?
39. Är konsolens teckenuppsättning i C/C++ samma som i C/C++:s grafiska miljöer?
40. Varför valde vi ett *Java*-program för att få tag i Unicode-koderna till några svenska, grekiska och arabiska tecken? Hade det varit möjligt att göra samma sak med ett C/C++ program?
41. Vad är koden för radfortsättning i C/C++ program? Skriv ett kort program för detta och testa det.
42. Varför kallas satsen **`float tal(0.5);`** i C++ för objektorienterad initiering?
43. Nämn två fördelar av objektorienterad initiering.
44. Vad innebär *Type system unification* och har den genomförts i C/C++?
45. Vad innebär det att C/C++ är *strikt typbestämta* programmeringsspråk?
46. I vilka två sammanhang genomförs *automatisk typkonvertering* i C/C++?
47. Beskriv med egna ord den s.k. *Tilldelningsregeln*. Testa den i programmet **`AssignRule`** på sid 56.
48. Ange värdet för variabeln **`s`** efter satserna:

```
int tal = 60000;
short s = tal;
```

Bekräfta ditt svar i ett litet C++ program. Förklara resultatet med någon regel.

49. Vad innebär talesättet ”slår runt” i typkonverteringssammanhang? När händert detta?
50. I vilket sammanhang tillämpas den s.k. ***int****-regeln*. Testa den i programmet **`IntRule`** på sid 59.
51. Vad innebär den s.k. *Befordringsregeln*. Testa den i programmet **`PromotInt`** på sid 61.
52. Ange värdet för uttrycket **`a*b`** efter satserna:
```
unsigned int a = 5;
short b = -3;
```
Bekräfta ditt svar i ett litet C++ program. Förklara resultatet med någon regel.
53. Vad är nackdelen med **`switch`**-satsen när man vill koda ett flervägsval? Vilka alternativ finns?
54. Ladda programmet **`TrickyElse`** (sid 64) i Visual Studio, kommentera bort måsvingarna **`{ }`** som i **`main()`** satts kring en enskild **`if`**-sats, kompilera och kör. Varför gör programmet inte det som det ska? Försök att korrigera det logiska felet utan att sätta tillbaka måsvingarna.
55. Hur borde enligt din åsikt flervägsvalet i Gissa tal-spelet optimalt kodas för att ge korrekt resultat?
56. Är enligt din åsikt alternativet med **`switch`**-satsen med tomma **`case`**-satser i programmet **`Switch-Inequ`** (sid 67) en bra lösning för flervägsval med olikheter?
57. Vad menas med *forward declaration* i C/C++?
58. Skriv en en **`do`**-loop där användaren får flera chanser att mata in två tecken i rätt ordning så länge de matas in i fel ordning. Programmet ska skriva ut **`OK`** om tecknen matats in i rätt ordning.

Kap 4 Fördjupning i C/C++ programmering, sid 76-106

59. Redogör för de tre olika betydelserna av hakparentesen **`[ ]`** i array-sammanhang och ge exempel.
60. **`int b[]`** innebär att **`b`** är en array av **`int`** och kan som parameter i en funktion ta emot en **`int`**-array när funktionen anropas. Hur kan koden skrivas i pekarversion?
61. Borde inte arrayens storlek anges i koden ovan (fråga 60) om **`b`** är en parameter i en funktion?
62. Är en array av **`char`** alltid en sträng i C/C++?
63. Vad är *nolltecknet* och vad används den för?
64. Vilken parameteröverföringsmetod används i C/C++ när parameterns datatyp är array?
65. Ger överskridning av indexgränserna i en array kompilerings- eller exekveringsfel?
66. Vad är en *referens* i C/C++? Vad är skillnaden mellan referens och pekare?
67. Rätta till felet i koden:
```
int  a = 5;
int& b = 6;
```
68. Redogör för de två olika betydelserna av ampersand **`&`** i referenssammanhang och ge exempel.
69. Varför kan man säga att referens en *automatiserad* pekare? Använd bilden på sid 88.
70. Skriv en pekarversion av programmet **`Array_Int`** (sid 77). Skapa arrayen med
```
int *no = new int[4];
```
och ersätt överallt *index* med *pekarposition*.
71. Vilken parameteröverföringsmetod används i C/C++ när parameterns datatyp är pekare?
72. Vad är den mest relevanta skillnaden mellan statisk och dynamisk minnesallokering?

73. Nämn de fem olika platser där datorn lagrar data när ett program kompileras och exekveras.
74. Är de platser du nämnde i frågan ovan (73) hårdvaru- eller mjukvarukomponenter?
75. Vilket är datorns snabbaste minne? Var är det lokaliserat? Skriv kod som skapar en variabal med snabbast möjliga åtkomst.
76. Med satsen **`auto float c;`** skapas en *automatisk* variabel. I vilken minnesenhet kommer denna variabel att lagras?
77. Blir det en statisk eller dynamisk minnesallokering med satsen **`auto float c;`** ? Vilket ord i satsen får utelämnas utan att det blir fel?
78. Vad heter den del av RAM som kompilatorn inte har tillgång till?
79. I vilken minnesenhet allokeras varaiabler som skapas med operatorn **`new`**? Blir det en statisk eller dynamisk allokering?
80. Vad heter den inversa operatorn till **`new`**? Vad gör den?
81. Om satsen **`static float b;`** står i en funktion kommer den lokala variabeln **`b`** att gälla i funktionen när den anropas, eller även leva vidare efter funktionsanropet?
82. Vilka processer körs i lagringsplatsen *Non-RAM* ?
83. Vad är nollpekaren? Hur skrivs den i kod? Vad kan den användas för?
84. Testa programmet **`Dynamic`** (sid 96) och förklara utgående från koden och med egna ord, vad som händer när det körs. Testa olika inmatningar, även med lite större textmängd.
85. Vad gör koden **`char s[8][6]`** ?

Kap 5 Filhantering, sid 107-133

86. Varför är filhantering relevant för programmering?
87. Nämn två operationer som står i centrum för filhantering?
88. Vilket bibliotek behövs för att kunna använda filhanteringsklasser i ett C++ program?
89. Vilken bibliotheksklass behövs för att skriva från ett C++ program till filer?
90. När och var någonstans i datorns fil- och mappsystem skapas filen **`Textfil.txt`** med satsen:

```
ofstream skrivFil("WriteRead.txt");
```

91. Vad händer om du exekverar satsen ovan (fråga 90) om filen **`WriteRead.txt`** redan finns?
92. Finns det i satsen ovan (fråga 90) en distinktion mellan objektet och referensen till objektet?
93. Skriv kod som behövs för att skriva texten **`"Hallo!"`** till filen som skapas i satsen ovan (fråga 90).
94. Skriv kod som stänger filen **`Textfil.txt`**. Vad händer när filen stängs efter användning?
95. Vilken klass behövs för att läsa från filer till ett C++ program?
96. Vad händer i satsen: **`ifstream lasFil("WriteRead.txt");`**
97. Vad heter funktionen i klassen **`ifstream`** som returnerar **`True`** när filslutstecknet läses?
98. Skriv kod som läser allt innehåll från filen som nämns i satsen ovan (fråga 96).

99. Modifiera satsen i fråga 90 för att lägga till text till filen **`WriteRead.txt`** utan att det gamla innehållet raderas.
100. Modifiera funktionen **`randPasswd()`** (sid 113) genom att implementera en ny lösenordpolicy för slumplösenord: 3 små bokstäver, 2 stora bokstäver och 2 specialtecken. Inkludera tecknen ? och @ i de stora bokstäverna, för att de är ”grannar” i ASCII-tabellen.
101. Testa den nya policyn i frågan ovan (100) för att skriva ut de nya slumplösenorden samt tillhörande användarnamn till en fil.
102. Modifiera funktionen **`krypt()`** (sid 118) genom att implementera en ny krypteringsmetod: Definiera krypteringen med funktionen $y = k\,x + m$, och anropa den med $k = 3$ och $m = -40$. Ordna dekrypteringen med den *inversa* funktionen $y = (x - m) / k$.
103. Testa den nya krypteringsmetoden i frågan ovan (102) och skriv den krypterade texten till en ny fil.

Kap 6 Introduktion till versionshantering, sid 134-143

104. Varför sysslar man med versionshantering inom systemutveckling, när den egentligen ligger utanför programmeringsämnet?
105. Hur växte versionshantering fram under informationsteknologins historia?
106. Vilka tre typer av versionshantering känner du till?
107. Av vilken typ är versionshanteraren *Git* bland de tre som efterfrågas i frågan ovan (106)?
108. Vilket program var föregångaren till Git?
109. Under utvecklingsarbetet av vilken känd produkt inom IT uppstod Git som en slags biprodukt?
110. Nämn tre viktiga verktyg som följer med när man installerar Git.
111. Vad står Bash för i verktyget Git Bash? Vad påminner namnet om?
112. Vad betyder Shell i Unix? Vad används det till?
113. Vilka typer av kommandon kan man (ur syntaxsynpunkt) skriva i Git Bash?
114. Vad är Unix-kommandot för att lista ut innehållet i en mapp?
115. Varför är konfigurationen av användarnamnet och mailadressen obligatoriska i Git?
116. Vad är ett *Git Repository*?
117. Var någonstans och med vilket git-kommando skapas ett nytt repository (repo)?
118. Skriv ett fullständigt kommando för att lista ut innehållet i repomappen i frågan ovan (117).
119. Med vilket git-kommando kan man kolla på läget i sitt repo?
120. Med vilket git-kommando kan man lägga en källkodsfil till repos index för versionshantering?

DEL II *Python, sid 169-286*

121. Koden **`int a`** ger i Python ett felmeddelande, medan **`a = 5`** är korrekt. Vilka slutsatser drar du av detta?
122. Varifrån får variabeln **`a`** i den korrekta satsen ovan (121) sin datatyp? Är datatypen av betydelse i Python?
123. Vilken datatyp returneras av den inbyggda funktionen `input()` ?
124. Med vilken pythonkod extraherar man bokstäver ur en sträng? Ge ett exempel.
125. Kan logiska variabler i Python tilldelas även 0 och 1?
126. Vad är skillnaden mellan **`random`** och **`random()`**?
127. Vilken typ av tal slumpar funktionen **`random()`** och i vilket intervall hamnar slumptalen? Ange det störst och det minst möjliga.
128. Skriv ett program som läser in tre heltal, med hjälp av enkla `if`- satser hittar och skriver ut det *största* av dem. Modifiera programmet så att det blir det *minsta* av de tre talen.
129. Modularisera programmet i frågan ovan (130) genom att flytta relevanta delar av koden till en pythonfunktion.
130. Skriv en `while`-sats som skriver ut de 50 första positiva heltalen. Skriv om loopen till en `for`-sats.
131. Skriv en `for`-sats som beräknar och skriver ut summan 1 + 2 + 3 + … + 100. Skriv om loopen till en `while`-sats.
132. Beskriv algoritmen *Primtalsfaktorisering* (sid 264) med egna ord. Varför är den rekursiv?
133. På vilken rad i koden ser man att `faktorisera()` (sid 264) är en rekursiv funktion?
134. Beskriv *Fibonaccis rekursionsformel* (sid 268) med egna ord.
135. Beräkna manuellt *F(4)* med Fibonaccis rekursionsformel.
136. Hur många kaninpar kommer det enligt rekursionsformeln att finnas efter *4* månader? Överensstämmer det med tabellen på sid 267?
137. Varför är Fibonaccis rekursionsformel rekursiv?
138. Hur kan man i koden se att `fib()` (sid 269) är en rekursiv funktion?
139. Vilket variabelnamn har den formella parametern i funktionen `fib()`?
140. Vilket variabelnamn har den aktuella parametern i `fib()`?
141. Vad är den praktiska nackdelen med det rekursiva programmet `Fibonacci` (sid 100) ?
142. Kan man koda Fibonacci även med en iterativ algoritm?
143. Skriv en rekursiv version av den iterativa Collatz-algoritmen.
144. Vilket externt bibliotek innehåller verktyg för att rita funktionsgrafer i Python?
145. Med den externa bilioteksfunktionen `plot()` kan man rita grafer till funktioner och t.ex. lösa ekvationer grafiskt. Programmet `Krimi_num` (sid 284) implementerar en algoritm som preciserar den onoggranna grafiska lösningen. Beskriv med egna ord denna algoritm *(Intervallhalveringsmetoden)*.