2016-03-23.kl.14-19

Tentaupplägg

TIPS 1:

Läs igenom ALLA uppgifterna. Välj den du känner är lättast först. Det kan gärna ta 10-20 minuter. Försök skriva saker som kan vara problem i uppgifterna. Är det något du absolut kommer att fastna på så kanske det är fel uppgift att ge sig på. Tiden du lägger på att noga läsa uppgifterna tjänar du in på att välja rätt uppgift.

TIPS 2:

Kolla ibland till kommunikationsfönstret. Det kan ha kommit information till alla utan att ni skickat in en fråga. Kanske gäller det dig också (d.v.s. den uppgift du jobbar med).

TIPS 3:

Om ni har problem med kompilator, Emacs eller annat som INTE har med uppgifterna att göra, räck upp handen så kommer en assistent. Detsamma gäller om hur man kopierar givna filer
" cp given_files/* . " eller liknande.

Frågor om själva uppgifterna tar vi i första hand via tentasystemet.

I körexemplen har vi markerat det som användaren matar in på tangentbordet med ***fet och kursiverad*** stil. Tänk på att körexemplen bara är *ett* exempel på när programmet körs. Testa ditt program noga och tänka över hur programmet skall fungera vid andra indata.

Vi hinner normalt sett inte svara på frågor de sista 10 utminutera på tentan. Då ägnar vi all tid åt att rätta uppgifter för att alla skall hinna få svar innan ni går hem.

Betygsgränser:	**Tid**	**DI/EL**	**SVP**
1 poäng	19:00	Betyg 3	Betyg G
3 poäng	18:00	Betyg 4	
3 poäng	16:45		Betyg VG
3 poäng	16:30	Betyg 5	
4-6 poäng	18:00	Betyg 5	
4-6 poäng	18:15		Betyg VG

OBS: Delpoäng delas inte ut på uppgifterna. För att få poäng på en uppgift måste man alltså lösa uppgiften helt (och enligt specifikation).

Lycka till med tenterandet och hoppas att alla får G på minst en uppgift idag.
M.v.h.
/Torbjörn (examinator)

2016-03-23.kl.14-19

Uppgift 1 - Portkod [1p]

Vi tänker oss en knappsats för portkoder som har 10 knappar. En tämligen osäker kod är en kod som bara innehåller en siffra. Om vi antar att man inte behöver trycka på någon "öppna"- eller "bekräfta"-knapp så är det ju bara att trycka på alla 10 knappar (i någon ordning) så kommer ju dörren att gå upp. Vi tänker oss därför en mycket säkrare variant där vi använder en tvåsiffrig kod. Man kan då fundera på hur många knapptryckningar som behövs för att få upp dörren. Lite snabbt inser man ju att det finns 10*10 = 100 kombinationer och att man då inte borde behöva fler än 200 tryckningar. Om man tänker en stund till så inser man ju att man inte ens behöver trycka så många gånger. Om man t.ex. trycker 0, 0 så kan man sedan trycka 1 och har då testat både koden 0, 0 och 0, 1. Därefter kan man trycka 0 igen för att också testa koden 1, 0. Då har man ju testat 3 koder men bara tryckt 4 gånger.

Skriv ett program som läser in siffror från standard inmatningsström tills användaren trycker enter och ctrl-d. Sekvensen av siffrorna motsvarar hur någon trycker på knappsatsen. Ditt program skall ta reda på hur många av de 100 tvåsiffriga kombinationer som sekvensen testar.

KRAV: Ditt program skall på något sätt hålla reda på vilka koder som är testade.

Körexempel:

```
Mata in en siffersekvens: 1 3 5 7 8 4 7 4 6 2 4 7 9 0 6 4 3 5 7
9 0 7 3 2 2 3 0 9 7 7
[ctrl-d]
Sekvensen täckte in 24 av 100 kombinationer.
```

2016-03-23.kl.14-19

Uppgift 2 - Filer och Kataloger [1p]

På datorn finns både filer och mappar (eller *kataloger* som de också heter). Vi tänker oss i denna uppgift att en katalog är precis som en fil, skillnaden är att en katalog inte innehåller egentligt data, utan bara en förteckning över andra filer. Vi skulle kunna tänka oss katalogen DIRECTORY.TXT:

```
erini02  3371 aug 11 14:41 banana.ali
torjo38  1394 aug 11 13:38 code.adb
erini02    52 aug 12 15:20 codecheck.adb
TDIU08   3344 aug 11 13:38 code.o
erini02 30776 feb  9 15:09 pidgey.bin
root      980 mar  2 08:27 passwords_in_clear.txt
```

Formatet är alltså först användaren som äger filen, sedan filens storlek (i byte, högerjusterat efter största filen i katalogen), sedan vilket datum och tid som filen sist ändrades (alltid 12 tecken) och sedan filens namn (max 128 tecken). Dessa data är separerade med blanksteg.

Din uppgift är att skapa ett program som hittar kopior av filer. Ditt program skall öppna filerna DIRECTORY.TXT och DIRECTORY2.TXT, gå igenom dessa och mata ut vilka filer som är kopior. Vi anser att två filer är kopior om de har exakt samma namn och exakt samma storlek.

KRAV: Du skall använda en struct för att lagra en fils egenskaper (d.v.s. användaren, storleken och filnmanet). Datum och tid behöver inte lagras.

Körexempel:

```
Följande kopior hittades i DIRECTORY.TXT och DIRECTORY2.TXT:
codecheck.adb
code.o
```

2016-03-23.kl.14-19

Uppgift 3 - Platser runt ett bord [2p]

Arnold skall ha lite gäster som skall placeras runt ett bord. När gästerna anländer så frågar han dem en i taget vilken plats de vill sitta på. Vissa gäster säger direkt att de kan sitta på en specifik plats (de är snälla nog att säga en ledig plats, ingen vill sitta i någon annans knä), men vissa är lite mer försiktiga och säger "Jag kan ta den platsen som blir över". Detta går bra, tycker Arnold, om inte fler börjar säga så, får då kommer sällskapet aldrig att få sätta sig till bords!

Skriv ett program där användaren får mata in hur många platser Arnold har runt sitt bord. Programmet skall sedan för varje person var den vill sitta. Det är givet att det är lika många platser runt bordet som personer, och att Arnold är en av dessa personer. Upp till en person kan vänta med sin placering till sist (då trycker användaren bara ENTER), och när alla andra har bestämt plats så får den personen den plats som blir över. Programmet skriver till sist ut en liten tabell över vem som fick vilken plats. Du behöver inte kontrollera att platser är lediga, eller att talen som matas in ligger i rätt intervall. Du måste däremot hålla koll så att inte fler personer börjar avvakta. Ditt program skall fungera även om ingen avvaktar. Funktionen *stoi* kan vara bra för att konvertera strängar till heltal.

Körexempel 1:

```
Hur många platser finns runt Arnolds bord: 5
Mata in plats för person 1: 3
Mata in plats för person 2: [ENTER]
Person 2 avvaktar.
Mata in plats för person 3: 1
Mata in plats för person 4: 2
Mata in plats för person 5: 5

Person | 1 | 2 | 3 | 4 | 5 |
-------+---+---+---+---+---+
Plats  | 3 | 4 | 1 | 2 | 5 |
```

Körexempel 2:

```
Hur många platser finns runt Arnolds bord: 4
Mata in plats för person 1: [ENTER]
Person 1 avvaktar.
Mata in plats för person 2: 4
Mata in plats för person 3: [ENTER]
En person avvaktar redan, du måste välja en plats!
Mata in plats för person 3: 3
Mata in plats för person 4: 1

Person | 1 | 2 | 3 | 4 |
-------+---+---+---+---+
Plats  | 2 | 4 | 3 | 1 |
```

I körexemplen ovan har alltså användaren tryckt på enter-tangenten för att visa att en person vill avvakta, man skall **inte** skriva in tecknen [ENTER].

2016-03-23.kl.14-19

Uppgift 4 - Måste fånga fler (Pidgeys) [2p]

En pidgey som du vill fånga.

Som alla vet blev Pokémon GO en stor succé sommaren 2016. Spelet som är ett s.k. AR-spel (Augmented Reality) går ut på att spelarna tar sig runt i den verkliga världen och träffar på och fångar fiktiva djur (s.k. Pokémon).
Pokémon GO går inte att *vinna* - precis som många andra stora nätbaserade flerspelarspel. Det går däremot att bli bättre genom att tjäna erfarenhetspoäng (XP). Efter ett visst antal XP är intjänade så går man upp en nivå (får s.k. level-up). Ju högre nivå du är, desto bättre - vilket för vissa är en fullt tillräcklig anledning att spela ett annars ganska "meningsfullt" spel.

Man kan få XP på många sätt, men ett av dem är att utveckla dina pokémon till nästa evolutionära stadie (detta kallas att "evolva" dem). Utöver den XP som man då tjänar får man också uppleva en fantastisk animation där ditt påhittade elektroniska djur blir ett annat påhittat elektroniskt djur. För att evolva dina pokémon så behöver du ge dem godis. Vissa pokémon kräver mindre godis att evolva och lämpar sig därför väl för att tjäna mycket XP, t.ex. pokémonen Pidgey. Här är några andra intressanta fakta:

- Man får alltid 3 godis när man fångar en Pidgey.
- Det kostar 12 godis att evolva en Pidgey.
- Man får alltid 1 godis när man evolvar en Pidgey.
- Man kan permanent bli av med en Pidgey genom att "ge bort den till professorn", men då får man åtminstone 1 godis i retur.

I denna uppgift skall du skriva ett program som räknar ut hur många Pidgeys man kan evolva, givet att man alltid ger bort den man just evolvat och eventuellt ger bort icke-evolvade Pidgeys för att kunna evolva fler.

Här är ett exempel. Nathalie har 10 st pidgey och 32 godis. Hur många kan hon evolva?
Utan att ge bort någon pidgey kan hon kan faktiskt evolva tre stycken. Hon evolvar två stycken (får 2 extra godis) och ger sedan direkt bort dem och får två godis till. Hon kan sedan evolva en tredje (som hon lika gärna också kan ge bort). Hon har nu 2 godis kvar och 7 Pidgeys. Hon kan inte ge bort så många som behövs för att evolva en till så då kan hon lika gärna behålla sina resterande Pidgeys till senare.

Körexempel:

```
Mata in hur många Pidgeys du har: 15
Mata in hur mycket godis du har från början: 45

Du kan evolva 5 Pidgeys.
Du har då 10 Pidgeys och 2 godis kvar.
```

OBS: Det är inte meningen att du skall ställa upp några matematiska ekvationer för att lösa detta. Låt ditt program helt enkelt testa sig fram till hur många Pidgeys som skall ges bort.