Föresläsning 15 - Examinationen

Idag ska vi tala om tentan, praktiska tentan, men också om laborationerna.

Först några allmänna ord om olika perspektiv på examination:

Målet med laborationerna och tentorna, ja, allt som ni ska redovisa är att ni ska få en **självständig kompetens**. Alla examinationsformer (alltså tentor och prov) är instrument som ska hjälpa oss att göra två saker:

1. Konstatera om en person har uppnått vissa kunskapsmål eller inte. Om målen är uppnådda får man godkänt betyg och kan gå vidare och utveckla sin förmåga på andra sätt. Det är här vi får meriter för poäng i LADOK. (Hurra!)

2. Utvärdera arten av måluppfyllelse för att skapa en "bild" eller kvalitativ beskrivning av den kompetens som ska utvecklas. (Vad betyder det då?)

Ettan kan vi förstå, det är en administrativ detalj som vi alla kan vara glada när den är uppfylld. Frågan här som är viktig är "klarade jag tentan eller inte?" och man vill ju förstås att svaret ska vara "JA".

Tvåan då, varför vill jag prata om den? Det står "Vad betyder det då?" där. Jo, tvåan är själva grunden till ettan egentligen, det är intressant att få svar på ettans fråga: "klarade jag tentan" men ännu intressantare är förstås frågan "hur klarade jag tentan?" eller om vi ställer frågan i rätt tidsperspektiv "vad ska jag göra idag för att ha stor chans att klara tentan?" Det klassiska svaret är ju då att man ska följa kursen och rent konkret i programmeringskursen är svaret att man ska arbeta med laborationer och övningsuppgifter som hör till tentan. Övningsuppgifterna hörande till tentan är publicerade på KTH-Social sedan ett tag och ni har arbetat med laborationerna under hela kursen. Så förhoppningsvis är alla på spåret, jag tror det.

Alla är troligen på spåret, som sagt, men det viktiga här, att säkerställa för mig, och er, är att ni arbetar på rätt sätt. I början av kursen sade jag att ni skulle hålla era laborationer hemliga för varandra. Det har inte varit praktiskt möjligt att göra det och vi kan förstå varför, jag har ju hela tiden sagt "arbeta med labbarna" och då är det klart att ni använt övningstid till det och suttit sida vid sida och då har ni förstås diskuterat med varandra om labbarna och det har då varit svårt att inte titta på varandras kod. Det syns ganska klart och tydligt i MOSS att samma kod förekommer i olika studenters program och det är OK *till en viss grad*, jag har ju gett exempel på kod som är bra att använda och det är då naturligt att den koden förekommer i olika studenters program. Men det blir problematiskt när kodavsnitt hörande till olika studenter uppvisar en exakt överensstämmelse. Det är fortfarande OK, men det är ett problem och jag vill som seriös utbildare hantera det problemet. Det betyder att jag ber några av er komma till mig för att prata om ert arbetssätt. Det är ingen fara att få en inbjudan till mig, det jag vill är att ge er ett extra stöd där vi säkerställer att ni arbetar och utvecklar en självständig kompetens. Några har fått detta mejl igår och ni är välkomna till mig som sagt. Se det som en extra möjlighet att komma till mig för att få mer hjälp.

Det här är också huvudsyftet med MOSS: att identifiera beroenden mellan studenter och trassla ut studenter ur för täta beroenden för att skolan ska kunna ge stöd till studenter att utveckla självständig kompetens. Jag vill likna det vid att kasta en krycka efter ett benbrott: om man bryter

ett ben behöver man extra stöd, man får kryckor och det blir lättare att gå med dessa kryckor, men målet är att gå självständigt. På samma sätt kan man få stöd när man programmerar i början, man kan titta på andras kod (jag hade tänkt mig att vi inte skulle titta på varandras labbkod, men det var tydligen svårt), att ta intryck av en annan students programkod är något som sker hela tiden men då det sker i väldigt stor utsträckning måste jag som examinator säga "stopp!" Det blir som att visa en läkare att man kan gå men ändå behöva kryckor. Så, de som får ett mejl till mig om detta är välkomna att besöka mig. Ni kommer att få en extra uppgift, troligtvis en tentauppgift, som befattar sig med samma område som ni laborerat på. Det innebär att ni inte tappar tid till studier inför själva tentorna. Ni tjänar på det hela i längden.

Låt oss nu låtsas att jag är en student som kommer och ska göra praktiska tentan...

[Rollspel följer, jag kommer att lösa någon eller några av uppgifterna av del A-uppgifterna som jag konstruerat.]

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

struct triangeltyp {
  float a, b, c;
};

skriv_triangel ( struct triangeltyp *t ) {
  printf("%.2f %.2f %.2f", t->a, t->b, t->c);
}

float area (struct triangeltyp t) {
  float p = (t.a+t.b+t.c)/2; //Bilda p-talet
  return sqrt(p*(p-t.a)*(p-t.b)*(p-t.c)); //Räkna ut och skicka tillbaks arean
}

main()
{
  //deklarera en array med 10 trianglar:
  struct triangeltyp triangelnr[10];
  int i, j;
  FILE * triangelfil;

  //Läs in trianglarna:
  triangelfil = fopen("trianglar.dat", "rb");
  for(i=0;i<10;i++)
  {
    fread(&triangelnr[i], sizeof(struct triangeltyp), 1, triangelfil);
  }
  fclose(triangelfil);

  //Skriv ut alla trianglar;
  for(i=0;i<10;i++)
  {
    skriv_triangel( &triangelnr[i] );
    printf(" Arean: %f\n", area(triangelnr[i]) );
  }
  printf("\n\n");
```

```
  //Sortera trianglarna på arean
  for(i=0;i<9;i++)
  {
    int minsta_triangelns_index=i;
    struct triangeltyp tmp;
    for(j=i+1;j<10;j++)
    {
      if( area(triangelnr[minsta_triangelns_index]) > area(triangelnr[j]) )
        minsta_triangelns_index = j;
    }
    //Sätt den hittills minsta triangeln på plats i:
    tmp = triangelnr[i];
    triangelnr[i]=triangelnr[minsta_triangelns_index];
    triangelnr[minsta_triangelns_index]=tmp;
  }

  //Skriv ut trianglarna igen:
  for(i=0;i<10;i++)
  {
    skriv_triangel( &triangelnr[i] );
    printf(" Arean: %f\n", area(triangelnr[i]) );
  }

  //Lägg dem i filen

  triangelfil = fopen("trianglar2.dat", "wb");
  for(i=0;i<10;i++)
  {
    fwrite(&triangelnr[i], sizeof(struct triangeltyp), 1, triangelfil);
  }
  fclose(triangelfil);


}
```

Nu kan det vara lämpligt att göra uppgift 2...

Vi tittar på en till uppgift, nr 5:

```
#include <stdio.h>
#include <math.h>

struct kk_triangel {
  float a,b;
};

skriv_kk_triangel (struct kk_triangel kkt) {
  printf("a: %.2f b: %.2f", kkt.a, kkt.b);
}

struct kh_triangel {
  float k,h;
};

skriv_kh_triangel (struct kh_triangel kht) {
  printf("k: %.2f h: %.2f", kht.k, kht.h);
}
```

```
void omvandla(struct kk_triangel *kkt, struct kh_triangel *kht)
{

  //Vilket håll ska omvandlingen ske åt?
  // kkt -> kht ?
  if( kkt->a!=0.0 || kkt->b!=0.0)
  {
    //printf("Katet 1: %f.\n", kkt->a);
    //printf("Katet 2: %f.\n", kkt->b);
    kht->k = kkt->a;
    kht->h = sqrt( (kkt->a)*(kkt->a)+(kkt->b)*(kkt->b));
  }

  // kht -> kkt ?
  if( kht->k!=0.0 || kht->h!=0.0)
  {
    //printf("Katet: %f.\n", kht->k);
    //printf("Hypotenusa: %f.\n", kht->h);
    kkt->a = kht->k;
    kkt->b = sqrt((kht->h)*(kht->h) - (kht->k)*(kht->k));
  }
};


main()
{
  struct kk_triangel kk_triangel1;
  struct kh_triangel kh_triangel1;

  //Testa kkt -> kht:
  printf("Mata in två kateter hörande till en kk-triangel.\n");
  printf("Katet 1: "); scanf("%f", &kk_triangel1.a);
  printf("Katet 2: "); scanf("%f", &kk_triangel1.b);

  kh_triangel1.k=0.0; kh_triangel1.h=0.0;
  omvandla(&kk_triangel1,&kh_triangel1);
  skriv_kk_triangel (kk_triangel1);
  printf("\n->\n");
  skriv_kh_triangel (kh_triangel1);
  printf("\n\n");

  //Testa kht -> kkt:
  printf("Mata in en katet och en hypotenusa hörande till en kh-triangel.\n");
  printf("Katet: "); scanf("%f", &kh_triangel1.k);
  printf("Hypotenusa: "); scanf("%f", &kh_triangel1.h); //Måste vara längre

  kk_triangel1.a=0.0; kk_triangel1.b=0.0;
  omvandla(&kk_triangel1,&kh_triangel1);
  skriv_kh_triangel (kh_triangel1);
  printf("\n->\n");
  skriv_kk_triangel (kk_triangel1);
  printf("\n\n");

}
```

Nu kan det vara bra att göra uppgift 6...