Programmering, grundkurs, 8.0 hp HI1024, TEN1

Fredagen den 2 mars 2012

Tentamen består av två delar, del A och del B. Del A innehåller 4 kryssfrågor på olika teman inom *C*-programmering. Varje fråga är uppbyggd på ett påstående. Påståendet är antingen *helt korrekt* och är då alltså SANT, eller så finns *minst ett allvarligt fel* i påståendet och då är alltså påståendet FALSKT. Man kryssar det man tror på och om man kryssar rätt får man 2 poäng. Kryssar man fel får man –2 poäng. Det är alltså riskabelt att chansa. Man har alltid alternativet att kryssa AVSTÅR som då ger 0 poäng. Del A kan inte ge mindre än 0 poäng totalt. (Om man har mer fel är rätt får man alltså inga negativa poäng.) Del B innehåller 13 traditionella frågor som ni ska svara skriftligt på. Totalt antal poäng är 32. Gränsen för E är 14 och gränsen för Fx är 12. KTHs allämnna regler för examination gäller. (Gränserna för E och Fx är sänkta med 2 poäng.)

Del A.

Påstående 1. Då vi vill skicka heltalsarrayer till funktioner som parametrar genom funktionsprototypen

```
funk (int arr[], int length).
```

så anger vi alltså arrayens namn i parametern `arr` och antal element i parametern `length`. Detta har fördelen att man kan öka arrayens storlek genom att i funktionen `funk` öka på parametern `length`.

[] SANT [] FALSKT [] AVSTÅR

Påstående 2. Om en array har 10 platser numreras dessa från 0 till 9. Om man försöker skriva till plats nr 10 (som ju inte finns) så försöker man skriva utanför det minnesområde som hör till arrayen. *C* upptäcker detta och utökar arrayens storlek med 1 så att plats 10 finns.

[] SANT [] FALSKT [] AVSTÅR

Påstående 3. Man kan läsa in ett innehåll i en sträng med `scanf` men då kan man inte läsa mellanslag eller tabulatorsteg, vi kan alltså skriva

```
char namn[20];
scanf("%s",namn);
```

och korrekt läsa in namn som inte innehåller mellanslag (eller tabulatorsteg). Om vi vill läsa in strängar som innehåller mellanslag, tabulatorsteg måste vi använda funktionen `gets()`. Denna funktion kan till och med läsa in strängar som innehåller nyradstecken.

[] SANT [] FALSKT [] AVSTÅR

Påstående 4. Förprocessordirektiv kan hjälpa oss att definera värden i ett *C*-program som ska gälla överallt i programmet. Vi kan då få en mer elegant programkod om detta kombineras med `switch`-satser.Vi kan skriva

```
#define MONDAY 0
#define TUESDAY 1
#define WEDNESDAY 2
#define THURSDAY 3
#define FRIDAY 4
#define WEEKEND 5
```

och i programmet ha koden

```
switch(day)
{
  case MONDAY: case TUESDAY: case WEDNESDAY: case THURSDAY: case FRIDAY:
    printf("Work on, no weekend yet...\n");
    break;
  case WEEKEND:
    printf("Weekend! No worries!\n");
    break;
}
```

(Här förutsätts `day` vara deklarerad på lämpligt sätt och programkoden förutsätts förekomma i ett lämpligt sammanhang i någon funktion, kanske `main()`. `#define`-direktiven finns längst upp i programmet.) Det finns en flexibilitet inbyggd här, om man till exempel i programmet vill tilldela andra värden till till exempel `MONDAY` så kan man skriva `MONDAY = 10;` (om `10` är det nya värde som vi vill att `MONDAY` ska stå för.)

[] SANT [] FALSKT [] AVSTÅR

Del B. (Nödvändiga `#include`-direktiv är ofta utelämnade, men ska anses finnas där ändå.)

Uppgift 11. (2p) Studera nedanstående program:

```
main()
{
  int i,j,p=0;

  for(i=1;i<=5;i++)
  {
    for(j=1;j<=5;j++)
      if (i%j!=0 && j%i!=0)p++;
  }
  printf("p: %d.\n", p);

}
```

Ange den utskrift som uppkommer då man kör det.

Uppgift 12. (2p) Studera nedanstående program:

```
main()
{
  int a, b, c;
  double x, y, z;

  a = 10; b = 20; c = 30;

  x = c/b;  y = a/c;  z = c/b;

  printf("%.2lf\n%.2lf\n%.2lf\n", x, y, z);

}
```

Ange den utskrift som uppkommer då man kör det. (Ledning: "`%.2lf`" är ett format för utskrift av långt flyttal, alltså `double`, med två decimaler.)

Uppgift 13. (2p) Studera nedanstående program:

```
main()
{
  int i, a[] = {0, 6, 5, 4, 7, 1, 2, 3};
  char s[] = "VALLGRAV";

  for(i=0;i<=7;i++)
    printf("%c ", s[a[i]] );
  printf("\n");
}
```

Ange den utskrift som uppkommer då man kör det.

Uppgift 14. (2p) En aritmetisk talföljd är en följd av heltal med samma avstånd mellan sig. Vi kan åstadkomma en sådan genom att bilda

```
main() {
  int i, a;
  for(i=0;i<4;i++) {
    a = 3*i  + 2;
    printf("%d ", a);
  }
  printf("\n");
}
```

Detta ger utskriften

```
2 5 8 11
```

Alltså en aritmetisk följd av tal med avståndet 3.

Modifiera programmet ovan så att vi skriver ut 10 tal och om talet som skrivs ut är jämnt ska det omringas av två stjärnor, en på var sida om talet, exempel: `*8*`.

Uppgift 15. (2p) Studera nedanstående program:

```
int f1(int a, int b)
{
  int w;
  a=b;
  b=a;
  w=a+b;
  printf("F1 %d %d\n", a, b);
  return w;
}

f2(int *j, int *i)
{
  int z;
  *j--; *i=*j-*i/2;
  z = f1(*i,*j);
  printf("F2 %d %d %d\n", *i, *j, z);
}

main()
{
  int i=10,j=20;
  f2(&i,&j);
  printf("MAIN %d %d\n", i, j);
}
```

Ange den utskrift som uppkommer då man kör det.

"*" Uppgift 16. (3p) Studera nedanstående program, det är tänkt att användas för beräkning av omvandlingar mellan två olika sätt att ange temperaturer, Celsius och Fahrenheit. Programmet har tre felaktiga rader. Finn dessa felaktiga rader och rätta dem

```
#include <stdio.h>

#define C_TO_F 0
#define F_TO_C 1

void convert (double *temperature, int mode)
{
  if(mode==C_TO_F)
  {
    *temperature = 9.0/5.0*(*temperature) - 17.0;
  }
  else
    *temperature = 9.0/5.0*(*temperature) - 17.0;
}

void input(double *temperature)
{
  printf("Temperature: "); scanf("%lf", temperature);
}

main()
{
  double temperature, temperature_copy;

  input(&temperature); temperature_copy = temperature;
```

```
  convert(&temperature, F_TO_C);
  printf("Temperature converted from Fahrenheit to Celsius: %.2lf.\n",
temperature);

  convert(&temperature_copy, C_TO_F);
  printf("Temperature converted from Celsius to Fahrenheit: %.2lf.\n",
temperature_copy);

}
```

Uppgift 17. (2p.) I *Del A* förekom flera falska påståenden. Välj två av dessa falska påståenden och förklara vad som egentligen gäller i sammanhanget, det vill säga uttryck vad som skulle vara sant i sammanhanget och varför det är på det viset.

Uppgift 20. (3p) Nedanstående program är tänkt att beräkna ett närmevärde på talet pi, om man tar tillräckligt många termer (styrs av `antal_termer`) i summan som beskrivs matematiskt i nedanstående program och beräknar rotuttrycket så får man ett värde som kommer närmare och närmare pi ju mer termer man tar. Dock finns två allvarliga fel i programmet. Beskriv dessa fel och ange hur man kan rätta dem.

```
main()
{
  int j, antal_termer;

  double sum, precision, pi;

  printf("Antal termer: "); scanf("%d", &antal_termer);

  for(j=1;j<=antal_decimaler;j++)
    sum+=1/(j*j);

  pi = sqrt(6*sum);

  printf("Pi: %lf.\n", pi);

}
```