KTH – STH

TENTAMEN

`HI1024:TEN1` - *Teoretisk tentamen*
Tid: Torsdagen den 20 oktober 2011, 8.15-12.15

Gamla kurskoder: HI1900, 6E2950, etc.
Examinator: *Johnny Panrike*
Rättande lärare: *Nicklas Brandefelt*, *Johnny Panrike* och *Peter Steiner*
Tentamen konstruerad av de som rättar tentan.

Tentamen består av **8** sidor inklusive denna sida. Längst bak finns en ASCII-tabell.

Instruktioner. Tentan har en **A**-del och en **B**-del. A-delen innehåller fem kryssuppgifter. B-delen innehåller 12 uppgifter som man ska svara skriftligt på. För att få minsta godkända betyg, E, ska man nå runt hälften av poängen. Uppgift 10, 11och 13 kräver fullständiga motiveringar för full poäng. Alla andra uppgifter kräver endast svar utan motiveringar. Uppgift 10 och 11 har högre poäng än de andra uppgifterna och det är för att motiveringarna ska också poängsättas. Uppgift 13 är en speciell uppgift. Sista uppgiften brukar vara lite speciell, om du inte förstår den kan du tryggt ägna dig åt de andra uppgifterna.

Tillåtna hjälpmedel: Inga alls!

Del A

Denna del innehåller 5 påståenden som antingen är helt korrekta eller innehåller minst ett allvarligt fel. Man ska kryssa det som gäller för att få poäng. Om man kryssar rätt får man 2 poäng. Om man kryssar fel får man –1 poäng, det är alltså riskabelt att chansa. Man kan inte få mindre än noll totalt på del A och max på del A är alltså 10 poäng. Man kan alltid avstå från att svara vilket säkert ger 0 poäng.

Påstående 1

Om man vill att en funktion som man anropar i *C* ska förändra innehållet i en variabel deklarerad utanför funktionen så går inte det på annat sätt än att man låter funktionen returnera ett värde med `return`.

[] Korrekt [] Felaktigt [] Avstår

Påstående 2

En lokal variabel deklarerad inne i en funktion kan ha samma namn som en lokal variabel deklarerad inne i en annan funktion. Dock kan man inte ha ett namn på en lokal variabel i en funktion om man har har det namnet på en parameter hörande till en annan funktion i samma program.

[] Korrekt [] Felaktigt [] Avstår

Påstående 3

Om en variabel eller funktion ska vara åtkomlig för en funktion måste den vara deklarerad ovanför den funktionen. För att två funktioner då ska kunna vara tillgänliga för varandra kan man *förhandsdeklarera* den ena funktioner och lägga ovanför den andra. Följande är alltså giltigt:

```
funk1();
funk2(){
   ...
}
funk1(){
   ...
}
```

Här kan båda funktionerna anropa varandra. Mellan måsvingarna finns det jobb som funktionerna gör, symboliserat med "`...`", överst ser vi alltså förhandsdeklarationen av funktionen `funk1()`.

[] Korrekt [] Felaktigt [] Avstår

Påstående 4

Om man har en *struct* som formell parameter till en funktion så kopieras, vid ett anrop, hela innehållet i den aktuella parametern. Detta kan ta mycket tid om structen är stor och det är då värt att endast skicka en pekare till den struct man önskar skicka till funktionen.

[] Korrekt [] Felaktigt [] Avstår

Påstående 5

Då man skickar en array av arrayer (också kallat matris) av till exempel heltal (`int`) till en funktion kan man inte deklarera det med dubbla hakparenteser av formen

```
funk( int param[][] )
```

man behöver här ange den så kallade *elementstorleken* för att funktionen ska kunna hålla reda på hur många heltal det är på varje rad i matrisen av heltal. En korrekt funktionsprototyp här skulle kunna vara

```
funk( int param[][8] )
```

om `param` ska ha 8 heltal i varje rad.

`[] Korrekt  [] Felaktigt  [] Avstår`

Del B

I många av uppgifterna är nödvändiga inkluderingsdirektiv som `#include <stdio.h>` och liknande utelämnade, uppgiften ska dock lösas som om direktiven skulle funnits med.

6. Vad ger följande program för utskrift:

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

  for(i=0;i<5;i++)
    for(j=i+1;j<5;j++)
      a++;

  printf("a: %d.\n", a);

}
```

(2p)

7. Vad ger följande program för utskrift:

```
main()
{
  int i, j;
  for(i=0;i<4;i++)
  {
    int z;
    for(z=4-i;z>0;z--)printf("   ");  //Tre mellanslag, bredd = 3 tecken
    for(j=-i+1;j<i;j++)
    {
      printf("%3d", j);  //Utskriftsbredd = 3 tecken
    }
    printf("\n");
  }
}
```

(3p)

8. Vad ger följande program för utskrift:

```
struct tretal {
  int a,b,c;
};

void f1( int *x, int *y, int z ) {
  int i;
  for(i=0;i<z;i++){ (*x)++; (*y)--; }
}

void oka(int a) {
  a = a+1;
}

int f2( struct tretal *w ) {
  oka(w->a);  oka(w->b);  oka(w->c);
  f1(&(w->a),&(w->c),w->c);
  return w->a + w->b + w->c;
}

main() {
  struct tretal t1;
  t1.a = 1;  t1.b = 3;  t1.c = 5;
  int tal;

  tal = f2(&t1);
  printf("%d: %d, %d, %d.\n", tal, t1.a, t1.b, t1.c);
}
```

(4p)

9. Vad ger följande program för utskrift:

```
int vokal(char ch) {
  if(ch>='A' && ch<='Z')ch+=32;
  if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' || ch=='y')
    return 1;
  return 0;
}

byt(char str[]) {
  int i, n=strlen(str);
  for(i=0;i<n;i++)if(vokal(str[i]))str[i]='i';
}

main() {
  char str1[] = "HAHAHOHOHEHE hahahohohihi";

  byt(str1);
  printf("%s\n", str1);
}
```

(3p)

10. Det finns tre problem med nedastående program som vill skriva ut tecken från en array av tecken... finn dessa problem och föreslå rättelser. Fullständiga motiveringar krävs för denna uppgift.

```
const int data[] = {2,4,6,8,1,3,5,7};
const char alfabetet[] = "abcdefghijklmnopqrstuvwxyz";

skriv_ut_tecken(const char teckenarray[], const int positioner[], int antal) {
  int i;
  for(i=0;i<antal;i++)
    printf("%s ", teckenarray[positioner[i]]);
}

main()
{
  int i, j;
  skriv_ut_tecken(alfabetet,data[0],15);
  printf("\n");
}
```

Ledning: Att det står `const` lite här och var har vi inte studerat så noga, men det är inte det som är problemet i det här programmet så du kan bara bortse från det här.

(5p)

11. Antag att du har en struct som är deklarerad så här

```
struct lager_status
{
  int varu_id;
  int antal;
  float pris;
};
```

Detta bygger på att vi har flera varor numrerade från 0 och uppåt med `varu_id`. Vi vill nu skriva en funktion som tar fram det totala värdet av alla varor med ett visst varu-id ur lagret. Ett *lager* beskrivs i något sammanhang av en array av structar av denna typ. Vi tänker oss då en funktion som opererar på en array av sådana structar som ser ut så här:

```
float total_varde_av(int varu_id, struct lager_status s[], int antal)
{
  int i; float buf_varde = 0.0;
  for(i=0;i<antal;i++)
    buf_varde = buf_varde + (s[i].pris)*(s[i].antal);
  return buf_varde;
}
```

Det finns dock ett problem med denna funktion... vilket är det? Föreslå en förbättring. Fullständiga motiveringar krävs för denna uppgift.

(5p)

12. Vad skriver följande program ut?

```
int a=4;

int func(int a)
{
  a++;
  return a;
}

int main(void)
{
  func(5);
  printf("%d",a);
}
```

(2p)

13. Varför blir inte utskrifterna identiska i följande exempel?

```
void func(int v[])
{
  printf("%ld\n",sizeof v);
}

int main(void)
{
  int v[4];
  printf("%ld\n",sizeof v);
  func(v);
}
```

(2p)

Ett utdrag ur ASCII-tabellen:

```
 32 : ' '  33 : '!'  34 : '"'  35 : '#'  36 : '$'  37 : '%'  38 : '&'
 39 : '''  40 : '('  41 : ')'  42 : '*'  43 : '+'  44 : ','  45 : '-'  46 : '.'
 47 : '/'  48 : '0'  49 : '1'  50 : '2'  51 : '3'  52 : '4'  53 : '5'  54 : '6'
 55 : '7'  56 : '8'  57 : '9'  58 : ':'  59 : ';'  60 : '<'  61 : '='  62 : '>'
 63 : '?'  64 : '@'  65 : 'A'  66 : 'B'  67 : 'C'  68 : 'D'  69 : 'E'  70 : 'F'
 71 : 'G'  72 : 'H'  73 : 'I'  74 : 'J'  75 : 'K'  76 : 'L'  77 : 'M'  78 : 'N'
 79 : 'O'  80 : 'P'  81 : 'Q'  82 : 'R'  83 : 'S'  84 : 'T'  85 : 'U'  86 : 'V'
 87 : 'W'  88 : 'X'  89 : 'Y'  90 : 'Z'  91 : '['  92 : '\'  93 : ']'  94 : '^'
 95 : '_'  96 : '`'  97 : 'a'  98 : 'b'  99 : 'c' 100 : 'd' 101 : 'e' 102 : 'f'
103 : 'g' 104 : 'h' 105 : 'i' 106 : 'j' 107 : 'k' 108 : 'l' 109 : 'm' 110 : 'n'
111 : 'o' 112 : 'p' 113 : 'q' 114 : 'r' 115 : 's' 116 : 't' 117 : 'u' 118 : 'v'
119 : 'w' 120 : 'x' 121 : 'y' 122 : 'z' 123 : '{' 124 : '|' 125 : '}' 126 : '~'
```