ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Kod:.....

Denna tentamen kommer att vara färdigrättad Ti 21/1 2003.
Lösningar till tentamensuppgifterna finns på nätet kl 13.00 idag.

Omtentamen i Programmeringsmetodik, 5p, Au2, D1 och E1, 030113

Hjälpmedel	: Inga	
Tid	: 08 -13	
Ansvarig lärare	: Gunnar Joki	Tel arb: 303317
		Tel hem: 274825

Svaren till uppgifterna 1-15 ska skrivas på tillgängligt utrymme i detta häfte. Behöver du mera utrymme kan du skriva på baksidan eller på extra papper. Lösningarna till uppgifterna 16-19 ska skrivas på utdelat extra papper med maximalt en uppgift per papper. Skriv din kod på varje inlämnat extra papper.

Den maximala poängen för respektive uppgift står angiven efter uppgiftens nummer. Totalt kan 40 poäng erhållas. För godkänt krävs ca 20 , för betyget 4 ca 28 och för betyget 5 ca 34 poäng.

Om inget speciellt anges gäller frågorna Borland C .

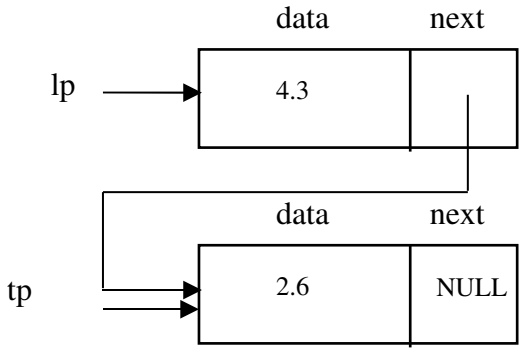
Detta häfte inlämnas.

Lycka till!

1) (1p) Antag att du har två pekare ap och bp som pekar på var sitt minnesutrymme innehållande reella tal. Avallokera det minnesutrymme som ap pekar och sätt ap att peka på samma minnesutrymme som bp pekar på.

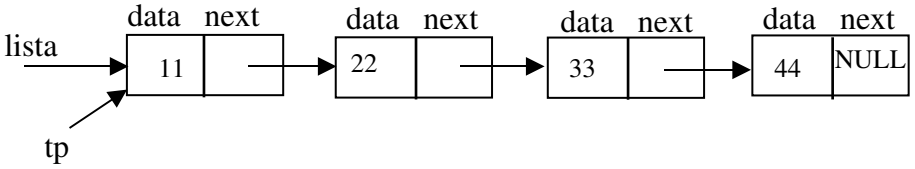
---

2)(1p) Skriv de satser som vänder på nedanstående länkade lista, som har länkar av linktyp. Pekaren lp ska alltså peka på länken vars data är 2.6 och denna länks nextpekare ska peka på länken med data 4.3 vars nextpekare ska NULL-ställas. Du får ej definiera några nya pekare.



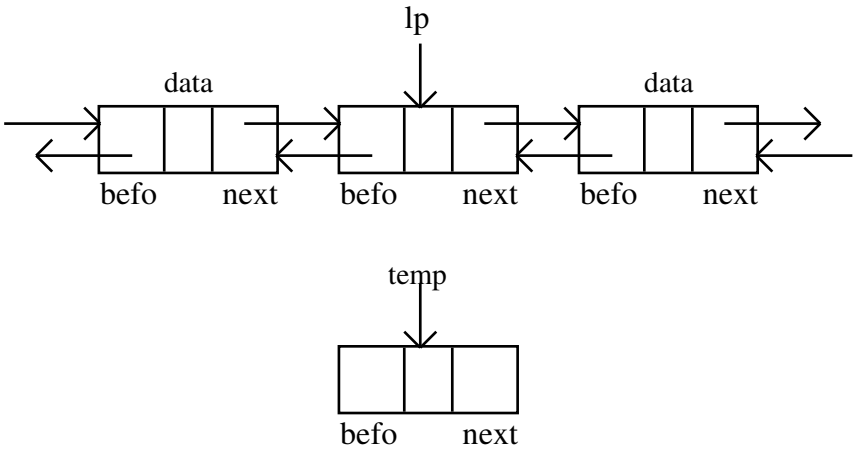
---

3) (1p) Skriv de satser som summerar data i en lista av nedanstående typ.



---

4) (1p) Skriv de satser som krävs för att byta plats mellan länken som temp pekar på och länken som lp pekar på i nedanstående tvåvägslista. Inga färdiga rutiner eller extra pekare får användas och listan ska hänga ihop efteråt.



5) (1p) Rita det binära träd som skapas då talen 23, 32, 13, 22, 33 och 15 instoppas tal för tal i trädet i den angivna ordningen, om instoppningsfunktionen sätter in data som är mindre i vänsterträdet och som är lika eller större i högerträdet. Söker man efter talet 15 i listan får man göra 6 jämförelser innan man hittat talet. Ange hur många jämförelser man får göra för motsvarande sökning i trädet.

---

6)(1p) Som index i en hashtabell för tal kan man exempelvis ta tal % 10. Skissa på hur en sådan hashtabell ser ut då talen ovan i uppgift 5 instoppas i den i angiven ordning om kollisioner hanteras med öppen adressering och hoppfunktionen hopp = 1 och sedan hopp = hopp + 1.

---

7)(1p) Hur kommer hashtabellen, enligt uppgift 6 ovan att se ut, om man istället använder stackar för att hantera kollisioner?

---

8)(1p) Vad blir utskriften från följande program?

```
#include <stdio.h>
void rf(int n)
{
   if (n > 0)
   {
      printf("%d ", n);
      rf(n-1);
      printf("%d ", n);
   }
}

void main()
{
   rf(3);
}
```

---

9)(1p) Antag att du har en 8 bitars unsigned char definierad enligt:

```
unsigned char uch = 7;
```

Ange värdet av uch efter satsen:

```
uch &= 4;
```

---

10)(1p)Antag att du har följande vektor av strängar:

```
char *str[] = {"Hej", "på" ,"dig"};
```

Vad är `str[0]`?

11)(2p)En term som $3.4x^2$ kan avbildas som en abstrakt datatyp enligt:

```
/* Term.h */

typedef
struct
{
   float koeff;   /* Koefficient ex 3.4 ovan */
   float expo;    /* Exponenten ex 2 ovan */
} term;

void las_term(term *tp);
/* Läser en term på formen ex 3.4x2 ovan */

void skriv_term(term t);
/* Skriver en term på formen ex 3.4x2 ovan */

int jfr_term(term t1, term t2);
/* Returnerar sant om t1.expo < t2.expo */

float value_term(term t, float x);
/* Returnerar värdet av termen för x */

term der_term(term t);
/* Returnerar termens derivata ex 6.8x1 ovan */
```

Implementera funktionen las_term.

---

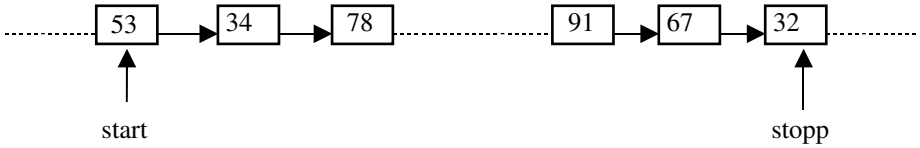
12)(2p)Implementera funktionen skriv_term ovan.

13)(2p)Fullborda funktionen utdata, så att den returnerar det heltal som finns i de fyra minst signifikanta bitarna av tal. Funktionshuvud enligt:

```
int utdata(int tal)
{
```

---

14)(2p)Antag att du har en envägslista , där länkarna är av samma typ, linktyp, som i uppgift 3 ovan med data i form av hela tal, enligt:



Fullborda funktionen search nedan, så att den söker efter data med värdet key i alla länkar från och med start till och med stopp. Finns key i denna del av listan returneras en pekare till länken ifråga, annars returneras NULL.

```
linktyp *search(linktyp *start, linktyp *stopp, int key)
{
```

---

15)(2p)Skriv en rekursiv funktion som åstadkommer samma sak som funktionen search i uppgift 14 ovan. Funktionshuvud enligt:

```
linktyp *rek_search(linktyp *start, linktyp *stopp, int key)
{
```

16)(5p)Skriv ett fullständigt program som börjar med att fråga efter antalet siffror för ett positivt binärt heltal och som dynamiskt skapar en sträng som talet ryms i. Programmet ska sedan läsa in talet till den skapade strängen, omvandla strängen till ett decimalt tal och skriva ut talet. Avslutningsvis ska det dynamiskt allokerade minnet frigöras. Ett körexempel där du matar in det understrukna:

```
Antal binära siffror? 4
Binärt tal? 1110
Talet decimalt = 14
```

---

17)(5p)Skriv ett fullständigt program som slumpar ett inläst antal kast med tärning och lägger dessa på en stack. Programmet ska sedan tömma stacken och räkna antalet ettor och antalet sexor. Avslutningsvis skrivs de procentuella förekomsterna av ettor och sexor ut. För hantering av stacken ska du använda:

```
/* Specifikation av LIFO-lista -- lifo.h */

typedef int datatyp;     /* Exempelvis */

typedef
struct link
{
  datatyp data;
  struct link *next;
} linktyp;

void push(linktyp **lpp, datatyp d);
/* Stoppar in d i LIFO-listan */

datatyp pop(linktyp **lpp);
/* Tar bort data från LIFO-listan */
```

---

18)(5p)Implementera funktionerna value_term och der_term, enligt uppgift 11 ovan och skriv ett huvudprogram som testar dessa funktioner genom att läsa in en term, derivera termen och skriva ut derivatan och dess värde för x = 2.0.

---

19)(5p)I textfilen term.txt finns ett antal termer av den abstrakta datatypen term, enligt uppgift 11 ovan, enligt:

```
5.6x2
-3.5x4
7.2x3
6.3x1
-1.5x3
. . . . .
. . . . .
```

Skriv ett fullständigt program som läser alla termer från filen och stoppar in dessa i en tvåvägslista sorterade i stigande ordning med avseende på exponenterna och därefter

förenklar listan genom att samla ihop termer med samma exponent. Avslutningsvis ska programmet skriva ut den förenklade listan Du ska även implementera funktionen jfr_term för den abstrakta datatypen term.
För hantering av tvåvägslistan ska du använda:

```
/* Specifikation av tvåvägslista -- twolist.h */

#include ”Term.h”
typedef term datatyp;

typedef
struct twolink
{
   enum {head, link} kind;
   struct twolink *befo, *next;
   datatyp data;
} headtyp, linktyp;

void newhead(headtyp **hpp);
/* Skapar en ny tom lista */

void newlink(linktyp **lpp);
/* Skapar en ny tom länk */

void putlink(datatyp d, linktyp *lp);
/* Sätter in data i en länk */

datatyp getlink(linktyp *lp);
/* Returnerar data från länk */

void inlast(linktyp *lp, headtyp *hp);
/* Sätter in länken sist i listan */

void infirst(linktyp *lp, headtyp *hp);
/* Sätter in länken först i listan */

void inpred(linktyp *lp, linktyp *ep);
/* Sätter in första länken före den andra */

void insucc(linktyp *lp, linktyp *ep);
/* Sätter in första länken efter den andra */

void insort(linktyp *lp, headtyp *hp,
            int (*is_less)(datatyp d1, datatyp d2));
/* Sätter in länken sorterad enligt is_less */

linktyp *firstlink(headtyp *hp);
/* Returnerar pekare till första länken i listan */

linktyp *lastlink(headtyp *hp);
/* Returnerar pekare till sista länken i listan */

linktyp *predlink(linktyp *lp);
/* Returnerar pekare till länken före */

linktyp *succlink(linktyp *lp);
/* Returnerar pekare till länken efter */

int is_link(linktyp *lp);
/* Returnerar 1 om länk annars 0 */
```

```
int empty(headtyp *hp);
/* Returnerar 1 om listan tom annars 0 */

int nrlinks(headtyp *hp);
/* Returnerar antalet länkar i listan */


void outlist(linktyp *lp);
/* Tar bort länken från listan */

void elimlink(linktyp **lpp);
/* Tar bort, avallokerar och NULL-ställer länken */

void clearhead(headtyp *hp);
/* Tar bort alla länkar från listan */

void elimhead(headtyp **hpp);
/* Eliminerar och NULL-ställer listan */
```

Lösningar till omtentamen i Programmeringsmetodik, 5p, 030113

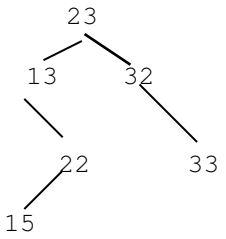
1)
```
free(ap);
ap = bp;
```

2)
```
tp->next = lp;
lp->next = NULL;
lp = tp;
```

3)
```
sum = 0;
tp = lista;
while (tp != NULL)
{
    sum += tp->data;
   tp = tp->next;
}
```

4)
```
temp->befo = lp->befo;
temp->next = lp->next;
lp->befo->next = temp;
lp->next->befo = temp;
lp->befo = lp->next = NULL;
```

5)



4 jämförelser

6)
```
 2 -> 32
 3 -> 23
 4 -> 13
 5 -> 22
 6 -> 33
 7 NULL
 8 -> 15
```

7)
```
 2 -> 22 -> 32
 3 -> 33 -> 13 -> 23
 4 NULL
 5 -> 15
```

8)
```
3 2 1 1 2 3
```

9)
```
4
```

10) Strängen "Hej"

11)

```
#include "Term.h"
#include <stdio.h>

void las_term(term *tp)
{
   printf("Term på formen 3.4x2 ? ");
   scanf("%fx%f", &tp->koeff, &tp->expo);
}
```

12)

```
#include "Term.h"
#include <stdio.h>

void skriv_term(term t)
{
   printf("%.2fx%.2f\n", t.koeff, t.expo);
}
```

13)

```
int utdata(int tal)
{
   return tal & 0xF;
}
```

14)

```
linktyp *search(linktyp *start, linktyp *stopp, int key)
{
   while (start != stopp && start->data != key)
   {
      start = start->next;
   }
   if ( start == stopp && stopp->data != key)
      return NULL;
   return start;
}
```

15)

```
linktyp *rek_search(linktyp *start, linktyp *stopp, int key)
{
   if (key == start->data)
      return start;
   else if (start == stopp)
      return NULL;
   else
      return rek_search(start->next, stopp, key);
}
```

16)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void main()
{
   char *sp;
   int i, antal, tal;

   printf("Antal binära siffror? ");
   scanf("%d", &antal);
   getchar();
   sp = calloc(antal+1, sizeof(char));
   printf("Binärt tal? ");
   gets(sp);
   tal = 0;
   for (i = 0; i < antal; i++)
   {
      tal = tal*2 + sp[i]- '0';
   }
   free(sp);
   printf("Tal = %d\n", tal);
   getch();
}
```

17)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "lifo.h"

void main()
{
   linktyp *lp = NULL;
   int antal, i, ettor = 0, sexor = 0, utfall;

   printf("Antal kast? ");
   scanf("%d", &antal);
   srand((unsigned)time(NULL));
   for (i = 1; i <= antal; i++)
   {
      utfall = rand() % 6 + 1;
      push(&lp, utfall);
   }
   while (lp != NULL)
   {
      utfall = pop(&lp);
      if ( utfall == 1)
         ettor++;
else if (utfall == 6)
         sexor++;
   }
   printf("Ettor = %.1f %%\n", ettor /(float)antal*100);
   printf("Sexor = %.1f %%\n", sexor /(float)antal*100);
   getch();
}
```

18)

```
#include <math.h>

float value_term(term t, float x)
{
   return t.koeff * pow(x, t.expo);
}

term der_term(term t)
{
   term temp;

   temp.koeff = t.koeff * t.expo;
   temp.expo = t.expo - 1;
   return temp;
}

#include "Term.h"
#include <conio.h>
#include <stdio.h>
void main()
{
   term t, d;

   las_term(&t);
   d = der_term(t);
   skriv_term(d);
   printf("%.2f", value_term(d, 2.0));
   getch();
}
```

19)

```
int jfr_term(term t1, term t2)
{
   return t1.expo < t2.expo;
}

#include <stdio.h>
#include <conio.h>
#include "Twolist.h"

void main()
{
   FILE *tsin;
   headtyp *hp, *nyhp;
   linktyp *lp, *nylp;
   term t, nyt;
   int nr, i, samma_exponent, akt_expo;
   float sum_koeff;

   tsin = fopen("Term.txt", "rt");
   newhead(&hp);
   while (fscanf(tsin, "%fx%f", &t.koeff, &t.expo)!= EOF)
```

```
      {
         newlink(&lp);
         putlink(t, lp);
         insort(lp, hp, jfr_term);
      }
      fclose(tsin);




      newhead(&nyhp);
      lp = firstlink(hp);
      while (lp != NULL)
      {
         t = getlink(lp);
         sum_koeff = t.koeff;
         akt_expo = t.expo;
         lp = succlink(lp);
         samma_exponent = 1;
         while (lp != NULL && samma_exponent)
         {
            t = getlink(lp);
            if (t.expo == akt_expo)
            {
               sum_koeff += t.koeff;
               lp = succlink(lp);
            }
            else
               samma_exponent = 0;

         }
         nyt.koeff = sum_koeff;
         nyt.expo = akt_expo;
         newlink(&nylp);
         putlink(nyt, nylp);
         inlast(nylp, nyhp);
      }
      elimhead(&hp);
      nylp = firstlink(nyhp);
      while (nylp != NULL)
      {
         skriv_term(getlink(nylp));
         nylp = succlink(nylp);
      }
      getch();
}
```