ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Kod:.....

Omtentamen i Programmeringsmetodik, 5p, Au2, D1 och E1, 020823

Hjälpmedel	: Inga	
Tid	: 14-19	
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 .

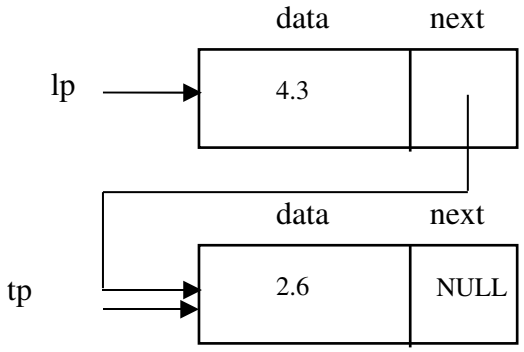
Detta häfte inlämnas.

Lycka till!

1) (1p) Antag att du har två pekare ap och bp som pekar på var sitt minnesutrymme innehållande reella tal. Skriv den sats som tilldelar det minnesutrymme som ap pekar samma värde som det som bp pekar på.

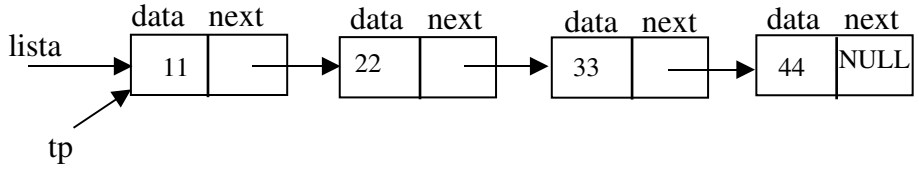
---

2)(1p) Skriv de satser som skapar en ny länk efter länken som har data 2.6 i nedanstående länkade lista som har länkar av linktyp. Pekaren tp ska flyttas så att den pekar på den nyskapade länken vars data ska vara 1.5 och vars nextpekare ska NULL-ställas. Du får ej definiera några nya pekare.



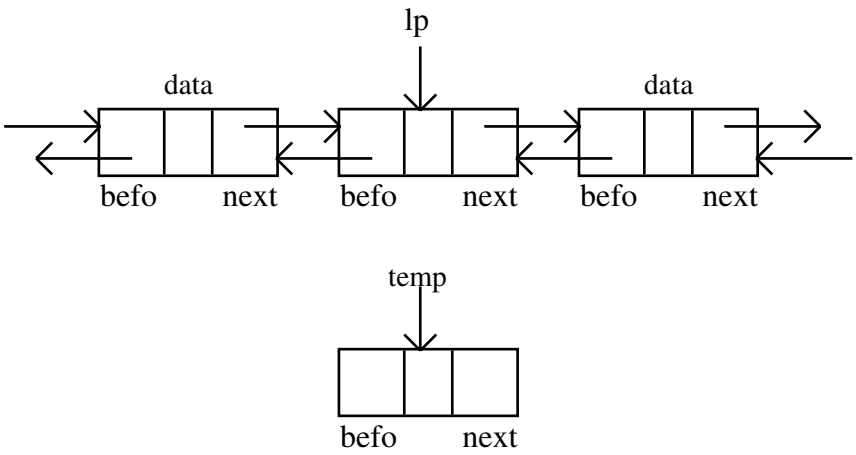
---

3) (1p) Skriv de satser som avallokerar den andra länken med data 22 i nedanstående länkade lista. Listan ska hänga ihop efteråt. Inga extra pekare får användas.



---

4) (1p) Skriv de satser som krävs för att stoppa in länken som temp pekar på före (till vänster om) 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 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.

---

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)
{
   printf(”%d ”, n);
   if (n > 0)
   {
      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 = 0x0F;
```

Ange det decimala värdet för uch efter satsen:

```
uch |= (1 << 4);
```

---

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

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

Vad är `str[2][2]`?

11)(2p)Ett mätresultat med mätvärde och mätfel kan avbildas som en abstrakt datatyp enligt:

```
/* Res.h */

typedef
struct
{
   float x;     /* Värde */
   float dx;    /* Absoluta felet */
} res;

void las_res(res *rp);
/* Läser ett resultat från tangentbordet */

void skriv_res(res r);
/* Skriver ett resultat på formen x +- dx */

int jfr_res(res r1, res r2);
/* Returnerar sant om r1.dx < r2.dx */

res add_res(res r1, res r2);
/* Addserar r1 och r2 */

res mul_res(res r1, res r2);
/* Multiplicerar r1 och r2 */
```

Implementera funktionen las_res.

---

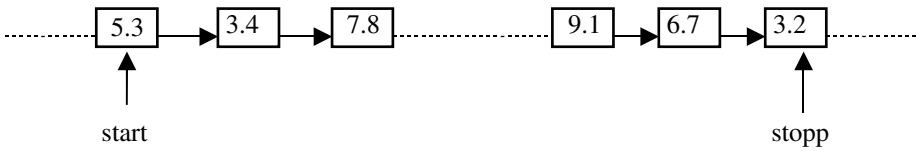
12)(2p)Implementera funktionen skriv_res, enligt uppgift 11 ovan.

13)(2p)Fullborda funktionen negativ, som ska returnera sant om tal är negativt annars falskt. Inga jämförelseoperatorer finns tillgängliga för tal, så du måste kontrollera om den mest signifikanta biten är ettställd, då är talet negativt. Du kan anta att int är 32 bitar. Funktionshuvud enligt:

```
int negativ(int tal)
{
```

---

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



Fullborda funktionen nolla nedan, så att den nollställer data för alla länkar från och med start till och med stopp.

```
void nolla(linktyp *start, linktyp *stopp)
{
```

---

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

```
void rek_nolla(linktyp *start, linktyp *stopp)
{
```

16)(5p)Skriv ett fullständigt program som börjar med att fråga efter antalet siffror för ett positivt 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 heltal och skriva ut talet. För omvandlingen får du använda funktionen atoi för vilken hjälpen ger följande information:

Syntax
#include <stdlib.h>
int atoi(const char *s);

Return Value
atoi returns the converted value of the input string. If the string cannot be converted to number of the corresponding type (int), atoi returns 0.

Avslutningsvis ska det dynamiskt allokerade minnet frigöras. Ett körexempel där du matar in det understrukna:

```
Antal siffror? 3
Tal? 213
Tal = 213
```

---

17)(5p)Skriv ett fullständigt program som upprepat slumpar positiva heltal mellan 0 och 9 så länge talet ej är 0 och lägger dessa på en stack. När 0 slumpas ska programmet tömma stacken och skriva ut alla tal som är större än det inlagda talens medelvärde. 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 add_res och mul_res, enligt uppgift 11 ovan och skriv ett huvudprogram som testar dessa funktioner. Funktionen add_res ska addera värdena och felen och mul_res ska multiplicera värdena och addera de relativa felen dx/x. Testprogrammet ska läsa in tre mätresultat a, b och c samt skriva ut resultatet av a + b*c.

19)(5p)I textfilen res.txt finns ett antal mätresultat av den abstrakta datatypen res, enligt uppgift 11 ovan, med värde och fel radvis enligt:

```
5.67 0.07
5.54 0.12
5.72 0.11
5.63 0.09
5.59 0.08
. . . . .
. . . . .
```

Skriv ett fullständigt program som läser alla resultaten från filen och stoppar in dessa i en tvåvägslista sorterade i stigande ordning med avseende på felen och därefter frågar efter hur många mätresultat som ska tas med i en medelvärdesberäkning och väljer ut detta antal mätresultat med de lägsta felen. Avslutningsvis ska programmet skriva ut det beräknade medelvärdet med felangivelse på skärmen. Felet i medelvärdet fås som summan av felen i de enskilda mätresultaten dividerat med antalet mätresultat. Du ska även implementera funktionen jfr_res för den abstrakta datatypen res.

För hantering av tvåvägslistan ska du använda:

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

#include ”Res.h”
typedef res 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, 020823

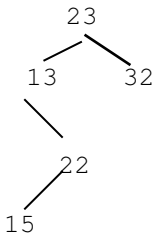
1)
```
*ap = *bp;
```

2)
```
tp->next = malloc(sizeof(linktyp));
tp = tp->next;
tp->data = 1.5;
tp->next = NULL;
```

3)
```
tp = tp->next;
lista->next = tp->next;
free(tp);
tp = NULL;
```

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

5)



6)
```
2 -> 32
3 -> 23
4 -> 13
5 -> 22
6 -> 15
```

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

8)
```
3 2 1 0 1 2 3
```

9)
```
31
```

10) Tecknet 'g'

11)

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

void las_res(res *rp)
{
   printf("Värde? ");
   scanf("%f", &rp->x);

   printf("Fel? ");
   scanf("%f", &rp->dx);
}
```

12)

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

void skriv_res(res r)
{
   printf("%.2f +- %.2f\n", r.x, r.dx);
}
```

13)

```
int negativ(int tal)
{
   if (tal & (1 << 31))
      return 1;
   return 0;
}
```

14)

```
void nolla(linktyp *start, linktyp *stopp)
{
   linktyp *lp = start;

   lp->data = 0.0;
   while (lp != stopp)
   {
      lp = lp->next;
      lp->data = 0.0;
   }
}
```

15)

```
void rek_nolla(linktyp *start, linktyp *stopp)
{
   if (start != stopp)
       rek_nolla(start->next, stopp);
   start->data = 0.0;
}
```

16)

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

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

   printf("Antal siffror? ");
   scanf("%d", &antal);
   getchar();

   sp = calloc(antal+1, sizeof(char));
   printf("Tal? ");
   gets(sp);

   tal = atoi(sp)

   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 tal, sum = 0, nr = 0;
   float medel;

   srand((unsigned)time(NULL));
   tal = rand() % 10;
   while (tal != 0)
   {
      nr++;
      sum += tal;
      push(&lp, tal);
      tal = rand() % 10;
   }
   if (nr > 0)
      medel = (float)sum / nr;
   while (lp != NULL)
   {
      tal = pop(&lp);
      if (tal > medel)
         printf("%d ", tal);
   }
   getch();
}
```

18)

```
res add_res(res r1, res r2)
{
   res r;

   r.x = r1.x + r2.x;
   r.dx = r1.dx + r2.dx;
   return r;
}

res mul_res(res r1, res r2)
{
   res r;

   r.x = r1.x * r2.x;
   r.dx = r.x * (r1.dx / r1.x + r2.dx / r2.x);
   return r;
}


#include "Res.h"
#include <conio.h>

void main()
{
   res a,b,c;

   las_res(&a);
   las_res(&b);
   las_res(&c);

   skriv_res(add_res(a, mul_res(b, c)));
   getch();
}
```

19)

```
int jfr_res(res r1, res r2)
{
   return r1.dx < r2.dx;
}

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

void main()
{
   FILE *tsin;
   headtyp *hp;
   linktyp *lp;
   res r, rsum = {0.0, 0.0}, rmed;
   int nr, i;

   tsin = fopen("Res.txt", "rt");
   newhead(&hp);
   while (fscanf(tsin, "%f%f", &r.x, &r.dx)!= EOF)
   {
      newlink(&lp);
      putlink(r, lp);
      insort(lp, hp, jfr_res);
   }
   fclose(tsin);

   printf("Antal  mätserier? ");
   scanf("%d", &nr);
   lp = firstlink(hp);
   for (i = 1; i <= nr; i++)
   {
      r = getlink(lp);
      rsum = add_res(r, rsum);
      lp = succlink(lp);
   }
   rmed.x = rsum.x / nr;
   rmed.dx = rsum.dx / nr;
   skriv_res(rmed);

   getch();
}
```