ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lösningarna till tentamensuppgifterna sätts ut på kurssidan på nätet under eftermiddagen.

Omtentamen i Programmeringsmetodik, 5p (1ED030), Au2, D1, Fri 2006-08-25.

Hjälpmedel	: Inga
Tid	: 08:00-13:00
Ansvarig lärare	: Christer Lindkvist 303393, 070-3273393

Svar till samtliga uppgifter 1-15 ska skrivas på utdelat extra papper. Använd ett papper till uppgifterna 1-5, två papper till uppgifterna 6-10 och ett papper per uppgift till uppgifterna 11-15 Skriv din tentamenskod på varje inlämnat extra papper.

Den maximala poängen för respektive uppgift står angiven i högermarginalen. Totalt kan 40 poäng erhållas. För betyget 3 krävs ca 20, för betyget 4 ca 28 och för betyget 5 ca 34 poäng.

Om inget speciellt anges gäller frågorna Visual C++.

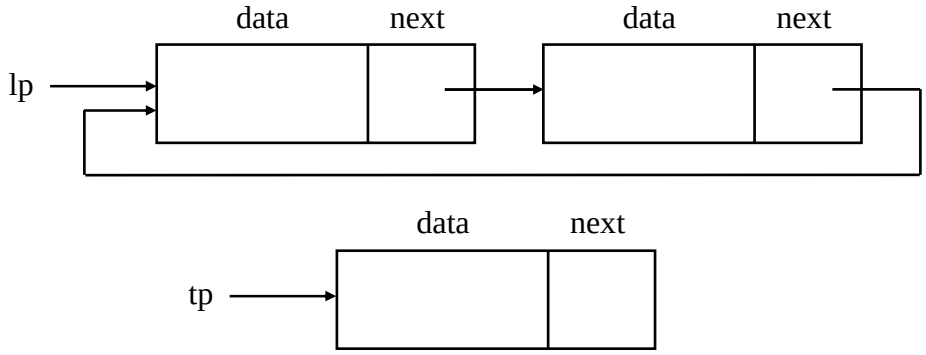
Detta häfte ska du behålla.

Lycka till!

1) Antag att du har en pekare bp som pekar på ett minnesutrymme som innehåller en bokstav. Skriv en sats som skriver ut denna bokstav på skärmen. **(1p)**

---

2) Stoppa in länken tp sist i den länkade strukturen nedan, dvs. efter den högra länken. Inga extra pekare får definieras och tp ska efter instoppning peka på samma länk som lp, alltså den första. Den sista structens nextpekare ska alltid peka på den första länken. **(1p)**



---

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

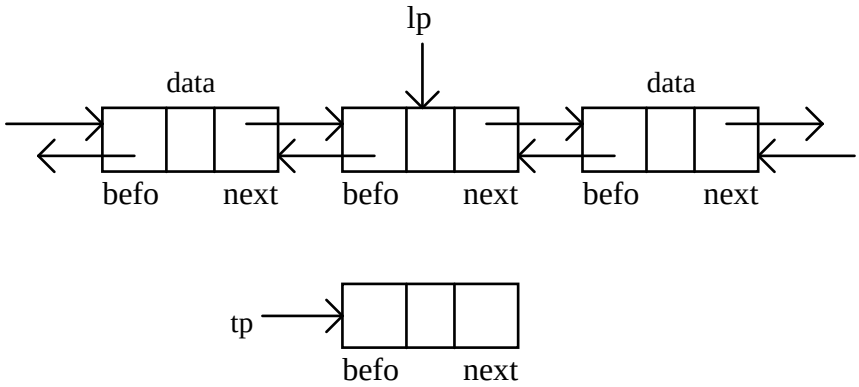
```
unsigned char uch = 0xf;
```

Ange värdet decimalt av uch efter satsen:

```
uch &= 21;
```

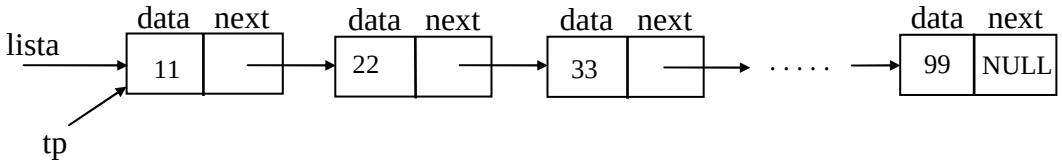
---

4) Ange hur du ställer om pekarna nedan så att länken tp instoppas före (till vänster om) länken lp i nedanstående tvåvägslista. Inga extra pekare får definieras och listan ska hänga ihop efteråt. **(1p)**



---

5) Skriv de satser som börjar med att leta upp den länk som har värdet 33 i nedanstående länkade struktur och som sedan ökar denna länks datavärde med datavärdet i länken som kommer efter. **(1p)**



---

6) Ett **positivt** allmänt bråk som exempelvis 5 2/3 kan avbildas som en abstrakt datatyp enligt (vi förutsätter för enkelhets skull att h, t och n alltid är positiva):

```
/* Specifikation av allmänt bråk -- allbrak.h */

typedef struct {
  int h;          /* Heltalsdelen, 5 i ex ovan */
  int t;          /* Täljare, 2 i ex ovan */
  int n;          /* Nämnare, 3 i ex ovan */
} allbrak;

void skapa_allbrak(allbrak *bp, int h, int t, int n);
/* Skapar ett allmänt bråk (h t/n). */

void las_allbrak(char *ledtext, allbrak *bp);
/* Skriver ut eventuell ledtext och läser därefter in ett */
/* allmänt bråk på formen h t/n från tangentbordet.       */

void skriv_allbrak(char *ledtext, allbrak b);
/* Skriver ut eventuell ledtext och ett allmänt bråk på */
/* skärmen. Utskriften är på formen h t/n.              */

void normera_allbrak(allbrak *bp);
/* Normerar bråket (förkortar och maximerar heltalsdelen) */

void ratnorm_allbrak(allbrak *bp);
/* Konverterar bråket till rationellt form (heltalsdelen = 0) */

allbrak add_allbrak(allbrak b1, allbrak b2);
/* Addition (b1 + b2) */

allbrak sub_allbrak(allbrak b1, allbrak b2);
/* Subtraktion (b1 – b2), vi förutsätter att b1 > b2 */

allbrak mul_allbrak(allbrak b1, allbrak b2);
/* Multiplikation (b1 * b2) */

allbrak div_allbrak(allbrak b1, allbrak b2);
/* Division (b1 / b2) */

int mindre_allbrak(allbrak b1, allbrak b2);
/* Jämförelse, sant om b1 < b2 */

int lika_allbrak(allbrak b1, allbrak b2);
/* Jämförelse, sant om b1 == b2 */
```

Du har även tillgång till dessa funktioner i uppgifterna nedan:

```
int gcd(int a, int b)    // Största gemensamma divisor
int lcm(int a, int b)    // Minsta gemensamma faktor
```

Implementera funktionerna **las_allbrak** och **skriv_allbrak**. Efter inläsningen av bråket ska det normeras med hjälp av **normera_allbrak** (se nedan). **(2p)**

---

7) Implementera funktionen **normera_allbrak** i uppgift 6 ovan. Vid normeringen skall bråket förkortas så långt som möjligt och heltalsdelen göras så stor som möjligt. Om täljaren är noll ska nämnaren vara ett (1 6/4 blir 2 1/2 och 1 6/3 blir 3 0/1). Vi förutsätter för enkelhets skull att h, t och n alltid är positiva. **(2p)**

---

8) Palindrom är ett ord (eller en mening) som har samma lydelse fram- och baklänges (som ”radar”, ”Naturrutan” och ”sirap i Paris”). Skriv en rekursiv funktion som returnerar sant (1) om en sträng är ett palindrom, annars falskt (0). Du kan förutsätta att strängen bara innehåller små bokstäver. Funktionen ska ha tre parametrar, en pekare till strängen samt index till första och sista tecknen som är kvar att undersökas. **(2p)**

```
int is_palindrom(char *str, size_t ind1, size_t ind2)
```

---

9) Fullborda funktionen **max_15**, som ska returnera sant (1) om uch ligger i intervallet 0 till 15, annars falskt (0). Antag att inga jämförelseoperatorer finns tillgängliga för uch och att funktionen därför måste kontrollera att alla de 4 mest signifikanta bitarna är nollställda. Funktionshuvud enligt: **(2p)**

```
int max_15(unsigned char uch)
```

---

10) Definiera en hashtabell för hela tal, som använder sig av en länkad lista av LIFO-typ för att hantera kollisioner och hashfunktionen *tal % 97*. Ange därefter på vilket index i hashtabellen som talet 120 hamnar. **(2p)**

---

11) Skriv ett program som dynamiskt allokerar en vektor som ska innehålla ett inläst antal allmänna bråk, enligt uppgift 6 ovan. Läs in talen till vektorn och skriv ut medelvärdet av talen i vektorn. Observera att även medelvärdet ska vara ett allmänt bråk. Därför måste summan och antalet tal vara det när medelvärdet beräknas. **(5p)**

---

12) Implementera funktionerna **skapa_allbrak**, **ratnorm_allbrak**, **add_allbrak** och **lika_allbrak**, för den abstrakta datatypen **allbrak** i uppgift 6 ovan. I **Skapa_allbrak** och **add_allbrak** ska bråket normeras. **Ratnorm_allbrak** ska omvandla bråket så att heltalsdelen blir noll (2 3/4 omvandlas till 0 11/4). Detta kan man exempelvis ha nytta av när man implementerar **lika_allbrak**! **(5p)**

---

13) Binärfilen **Allbrak.dat** innehåller ett antal allmänna bråk enligt uppgift 6 ovan. Skriv ett program som läser alla talen från binärfilen och placerar dessa i en kö. Programmet frågar sedan efter ett bråk (nyckel), tömmer kön och skriver ut alla tal som är mindre än nyckeln.

För hantering av kön ska du använda **fifo** nedan.

```
/* Specifikation av FIFO-lista -- fifo.h */

#ifndef FIFOH
#define FIFOH

#include "allbrak.h"
typedef allbrak datatyp;     /* Exempelvis */

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

void infifo(linktyp **lpp, datatyp d);
/* Stoppar in d i FIFO-lista */

datatyp utfifo(linktyp **lpp);
/* Tar bort data från FIFO-listan */

#endif
```

---

14) Skriv ett fullständigt program som slumpar fram ett inläst antal heltal och stoppar in dem i ett binärt träd. Därefter ska trädet skrivas ut på skärmen och en söknyckel läsas in. Efter sökning i trädet ska sökresultatet skrivas ut. För att hantera det binära trädet skall du endast använda den abstrakta datatypen **Tree** nedan. Utskriftsfunktionen **write_data** och jämförelsefunktionerna **is_equal** och **is_less** skriver du själv. **(5p)**

```
/* Specifikation av binärt träd -- tree.h */

#ifndef TREE_H
#define TREE_H

typedef int datatyp;

typedef struct nod {
  datatyp data;
  struct nod *left, *right;
} nodtyp;

void into_tree(nodtyp **rpp, datatyp d,
               int (*is_less)(datatyp, datatyp));
/* Stoppar in d i trädet ordnat enligt is_less */

void show_tree(nodtyp *rp, void (*write_data)(datatyp));
/* Skriver ut trädet med write_data funktionen */

nodtyp *search_tree(nodtyp *rp, datatyp key,
                    int (*is_equal)(datatyp, datatyp),
                    int (*is_less)(datatyp, datatyp));
/* Returnerar pekare till nyckelposten om den finns annars NULL */

int same_in_tree(nodtyp *rp, datatyp key,
                 int (*is_equal)(datatyp, datatyp),
                 int (*is_less)(datatyp, datatyp));
/* Returnerar antalet dataposter i trädet som är identiska */
/* med nyckelposten key.                                   */

#endif
```

15) Textfilen **Allbrak.txt** innehåller ett antal allmänna bråk på formen h t/n enligt uppgift 6 ovan. Skriv ett program som läser alla talen från textfilen och placerar dessa i en tvåvägslista sorterad efter storlek. Skriv därefter ut alla talen i listan. **(5p)**

För hantering av allmänna bråk ska du endast använda den abstrakta datatypen **allbrak** i uppgift 6 ovan. Komplettera **allbrak** med en funktion som läser bråk från en textfil och returnerar samma statuskod som fscanf:

```
int lasfil_allbrak(FILE *stream, allbrak *bp);
/* Läser in ett allmänt bråk på formen h t/n från textfil. */
/* Efter inläsning normeras bråket.                         */
```

För hantering av tvåvägslistan ska du använda:

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

#ifndef TWOLISTH
#define TWOLISTH

#include "allbrak.h"
typedef allbrak 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 fr†n listan */

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

#endif
```

Lösningar till tentamen i Programmeringsmetodik, 5p 2006-08-25

1)
```
printf("%c", *bp);
```

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

3)
```
    00001111
and 00010101
    00000101 = 5
```

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

5)
```
while (tp->data != 33)
  tp = tp->next;
tp->data += tp->next->data;
```

6)
```
#include <stdio.h>
#include "allbrak.h"

void las_allbrak(char *ledtext, allbrak *bp)
{
  if (ledtext) printf(ledtext);
  scanf("%d %d/%d", &bp->h, &bp->t, &bp->n);
  normera_allbrak(bp);
}

void skriv_allbrak(char *ledtext, allbrak b)
{
  if (ledtext) printf(ledtext);
  printf("%d %d/%d", b.h, b.t, b.n);
}
```

7)
```
void normera_allbrak(allbrak *bp)
{
  int sgd;

  bp->h += bp->t/bp->n;
  bp->t %= bp->n;

  if (bp->t == 0)
    bp->n = 1;
  else {
    sgd = gcd(bp->t, bp->n);
    bp->t /= sgd;
    bp->n /= sgd;
  }
}
```

8)
```c
int is_palindrom(char *str, size_t ind1, size_t ind2)
{
  if (ind1 > ind2)
    return 1;
  else if (str[ind1] != str[ind2])
    return 0;
  else
    return is_palindrom(str, ind1+1, ind2-1);
}
```

9)
```c
int max_15(unsigned char uch)
{
  int i;

  for (i = 4; i <= 7; i++) {
    if (uch & (1 << i))
      return 0;
  }
  return 1;
}
```

10)
```c
struct link {
  int data;
  struct link *next;
} *hashtabell[97];{
```

index = 120 % 97 = 23

11)
```c
#include <stdio.h>
#include <stdlib.h>
#include "allbrak.h"

void main()
{
  int i, nr;
  allbrak *bp, bsum, bnr;

  printf("Ge antal allmänna bråk: ");
  scanf("%d", &nr);
  bp = calloc(nr, sizeof(allbrak));
  skapa_allbrak(&bsum, 0, 0, 1);
  for (i = 0; i < nr; i++) {
    las_allbrak("Ge allmänt bråk (h t/n): ", &bp[i]);
    bsum = add_allbrak(bsum, bp[i]);
  }
  skapa_allbrak(&bnr, nr, 0, 1);
  skriv_allbrak("Medel = ", div_allbrak(bsum, bnr));
  free(bp);
}
```

12)
```c
void skapa_allbrak(allbrak *bp, int h, int t, int n)
{
  bp->h = h;
  bp->t = t;
  bp->n = n;
  normera_allbrak(bp);
}

void ratnorm_allbrak(allbrak *b)
{
  b->t += b->h*b->n;
  b->h = 0;
}

allbrak add_allbrak(allbrak b1, allbrak b2)
{
  allbrak sum;

  sum.h = b1.h + b2.h;
  sum.n = lcm(b1.n, b2.n);
  sum.t = (sum.n/b1.n)*b1.t + (sum.n/b2.n)*b2.t;
  normera_allbrak(&sum);
  return sum;
}

int lika_allbrak(allbrak b1, allbrak b2)
{
  ratnorm_allbrak(&b1);
  ratnorm_allbrak(&b2);
  return b1.t*b2.n == b2.t*b1.n;
}
```

13)
```c
#include <stdio.h>
#include "allbrak.h"
#include "fifo.h"

void main()
{
  linktyp *lp = NULL;
  FILE *bsin;
  allbrak key, b;

  bsin = fopen("allbrak.dat", "rb");
  fread(&b, sizeof(allbrak), 1, bsin);
  while(!feof(bsin)) {
    infifo(&lp, b);
    fread(&b, sizeof(allbrak), 1, bsin);
  }
  fclose(bsin);

  las_allbrak("Ge nyckelbråk (h t/n): ", &key);
  while (lp != NULL) {
    b = utfifo(&lp);
    if (mindre_allbrak(b, key)) {
      skriv_allbrak(NULL, b);
      skriv_allbrak(" < ", key);
      putchar('\n');
    }
  }
}
```

14)

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "tree.h"


void write_data(int data)
{
  printf("%6d", data);
}

int is_equal(int a, int b)
{
  return a == b;
}

int is_less(int a, int b)
{
  return a < b;
}


void main()
{
  nodtyp *rot = NULL;
  int i, antal, key;

  srand((unsigned) time(NULL));
  printf("Ge antalet slumptal: ");
  scanf("%d", &antal);
  for (i = 1; i <= antal; i++) {
    into_tree(&rot, rand(), is_less);
  }

  show_tree(rot, write_data); putchar('\n');

  printf("Ge nyckel: ");
  scanf("%d", &key);
  if (search_tree(rot, key, is_equal, is_less))
    printf("Talet %d finns i tädet!\n", key);
  else
    printf("Talet %d finns ej i tädet!\n", key);
 }
```

15) Komplettering av allbrak:

```
int lasfil_allbrak(FILE *stream, allbrak *bp)
{
  int status;

  status = fscanf(stream, "%d %d/%d", &bp->h, &bp->t, &bp->n);
  normera_allbrak(bp);
  return status;
}
```

Huvudprogram:

```
#include <stdio.h>
#include "allbrak.h"
#include "twolist.h"

void main()
{
   FILE  *tsin;
   headtyp *hp;
   linktyp *lp;
   allbrak b;

   newhead(&hp);
   tsin = fopen("Allbrak.txt", "rt");
   while (lasfil_allbrak(tsin, &b) != EOF)
   {
     newlink(&lp);
     putlink(b, lp);
       insort(lp, hp, mindre_allbrak);
   }
   fclose(tsin);

   lp = firstlink(hp);
   while (lp != NULL)
   {
      b = getlink(lp);
      skriv_allbrak("  ", b);
      lp = succlink(lp);
   }
   elimhead(&hp);
   putchar('\n');
}
```