ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lämna in ifylld kursvärdering tillsammans med tentamen!

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

Tentamen i Programmeringsmetodik, 5p (1ED030), Au2, D1, Fri 2006-05-22.

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++.

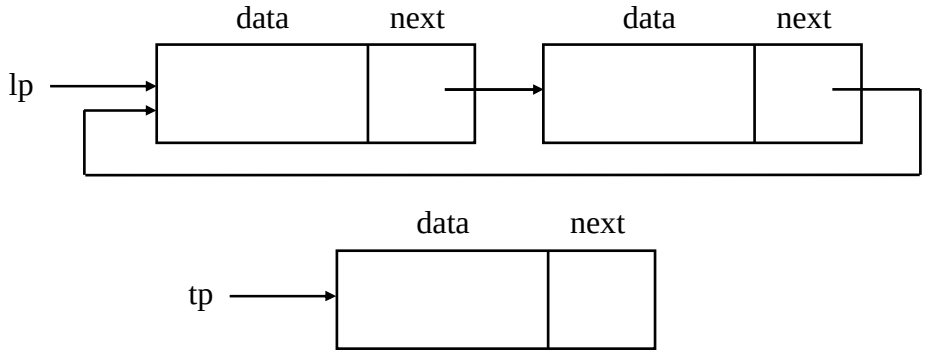
Detta häfte ska du behålla.

Lycka till!

1) Antag att du har en pekare bp som pekar på ett minnesutrymme som innehåller en versal (stor bokstav, A-Z). Skriv en sats som gör om bokstaven till motsvarande gemen (liten bokstav, a-z). **(1p)**

---

2) Stoppa in länken tp före länken lp i den länkade strukturen nedan. 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)**

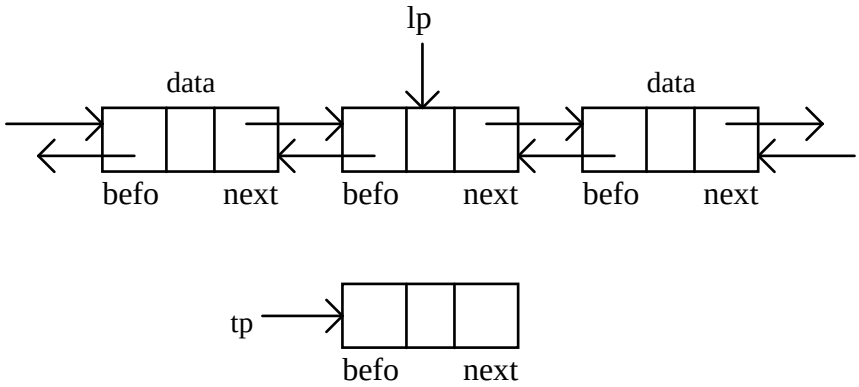
```
unsigned char uch = 0xf;
```

Ange värdet decimalt av uch efter satsen:

```
uch |= 21;
```

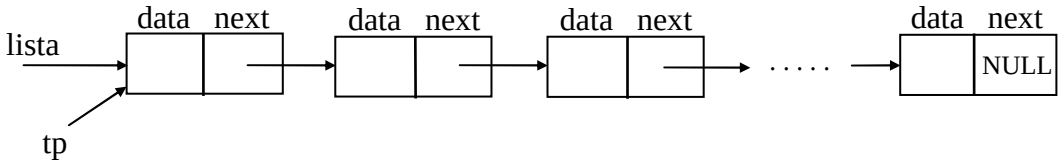
---

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



---

5) Skriv de satser som tar bort och avallokerar den sista länken (längst till höger) nedan. Inga extra pekare får definieras. Efter bortplockandet ska lista och tp peka på samma länk som före avallokeringen. **(1p)**



---

6) Ett rationellt tal, som exempelvis -2/3 kan avbildas som en abstrakt datatyp enligt:

```
/* Specifikation av rationella tal -- rtal.h */

typedef struct {
  int t;          /* Täljare (med tecken)      */
  int n;          /* Nämnare (normalt positiv) */
} rtal;

void skapa_rtal(rtal *rp, int t, int n);
/* Skapar och normerar ett rationellt tal (t/n) */

void las_rtal(char *prompt, rtal *rp);
/* Skriver ut eventuell ledtext och läser därefter in ett */
/* rationellt tal på formen t/n från tangentbordet, där t */
/* och/eller n kan vara negativa heltal. Efter inläsning  */
/* normeras talet.                                        */

void skriv_rtal(char *prompt, rtal r);
/* Skriver ut eventuell ledtext och ett rationellt tal på */
/* skärmen. Utskriften är på formen t/n.                  */

void normera_rtal(rtal *rp);
/* Normerar talet (förkortar och placerar tecknet i täljaren) */

rtal add_rtal(rtal r1, rtal r2);
/* Addition (r1 + r2) */

rtal sub_rtal(rtal r1, rtal r2);
/* Subtraktion (r1 – r2) */

rtal mul_rtal(rtal r1, rtal r2);
/* Multiplikation (r1 * r2) */

rtal div_rtal(rtal r1, rtal r2);
/* Division (r1 / r2) */

int mindre_rtal(rtal r1, rtal r2);
/* Jämförelse, sant om r1 < r2 */

int lika_rtal(rtal r1, rtal r2);
/* Jämförelse, sant om r1 == r2 */
```

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_rtal** och **skriv_rtal** så att inlästa (och utskrivna) tal blir normerade enl. ovan. Vid inläsning av exempelvis 4/-6 ska utskriften bli -2/3. **(2p)**

---

7) Implementera funktionen **normera_rtal** i uppgift 6 ovan. Vid normeringen skall bråket förkortas så långt som möjligt, nämnaren ska alltid vara positiv och ett negativt tal ska ha negativ täljare. Om täljaren är noll ska nämnaren vara ett. **(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 två parametrar, en pekare till första tecknet (som ska undersökas) i strängen och antal tecken som ska undersökas. **(2p)**

```
int is_palindrom(char *str, size_t n)
```

---

9) Skriv färdigt funktionen nedan så att den returnerar det tal som man får då man byter plats på de fyra mest signifikanta bitarna med de fyra minst signifikanta. **(2p)**

```
unsigned char swap_nibbles(unsigned char uch)
```

---

10) Som hashfunktion för strängar kan man exempelvis ta summan av ASCII-koderna för strängens tecken % 97. Skriv en hashfunktion som summerar ASCII-koderna för alla tecken i strängvariabeln str. På vilket index hamnar strängen "ABBA" i en tom hash-tabell, om man använder denna hashfunktion och bokstaven 'A' har ASCII-koden 65. **(2p)**

---

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

---

12) Implementera funktionerna **add_rtal**, **sub_rtal**, **mul_rtal** och **div_rtal**, för den abstrakta datatypen **rtal** i uppgift 6 ovan. **(5p)**

---

13) Skriv ett fullständigt program som läser in alla heltalspar (t n) från textfilen **Rtal.txt**, och skapar rationella tal (t/n) som läggs på en stack. Programmet frågar sedan efter ett ratinellt tal (nyckel), tömmer stacken och skriver ut alla tal som är större än nyckeln. Textfilen ser ut såhär, med ett talpar per rad: **(5p)**

```
 13   3
 -1   4
  2   5
-11  12
  …   …
```

För hantering av stacken ska du använda:

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

#ifndef LIFOH
#define LIFOH

#include "rtal.h"
typedef rtal 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 */

#endif
```

---

14) Skriv ett fullständigt program som gör ett inläst antal tärningskast, stoppar in alla tärningsutfall i ett binärt träd, skriver ut trädet, och räknar och skriver ut en tabell med antalet utfall (1-6). För att manipulera tärningen och det binära trädet skall du endast använda de abstrakta datatyperna **Dice** och **Tree** nedan. **(5p)**

```
/* Specifikation av tärning -- dice.h */

#ifndef DICE_H
#define DICE_H

typedef enum {UNDEFINED, ONE, TWO, TREE, FOUR, FIVE, SIX} dice;

void init_dice(dice *dp);
/* Initierar slumptalsgeneratorn och en tärning (om dp != NULL) */

void throw_dice(dice *dp);
/* Kasta tärningen och sparar utfallet */

void read_dice(dice *dp);
/* Läser tärningsutfall från tangentbordet */

void write_dice(dice d);
/* Visar tärningsutfallet på skärmen */

int get_dice(dice d);
/* Returnerar tärningsutfallet som ett heltal (1-6) */

void put_dice(dice *d, int u);
/* Sätter tärningsutfallet med ett heltal (1-6) */

int is_less_dice(dice d1, dice d2);
/* Jämförelse, sant om d1 < d2 */

int is_equal_dice(dice d1, dice d2);
/* Jämförelse, sant om d1 == d2 */

#endif
```

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

#ifndef TREE_H
#define TREE_H

#include "dice.h"
typedef dice 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) Binärfilen **Rtal.dat** innehåller ett antal rationella tal enligt uppgift 6 ovan. Skriv ett program som läser alla talen från binärfilen och placerar dessa i en tvåvägslista sorterad efter storlek. Använd funktionerna för den abstrakta datatypen **rtal** i uppgift 6 ovan. Skriv därefter ut alla rationella tal i listan som är större än noll. För hantering av tvåvägslistan ska du använda: **(5p)**

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

#ifndef TWOLISTH
#define TWOLISTH

#include "rtal.h"
typedef rtal 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 2006-05-22

1)
```
*bp = *bp – 'A' + 'a';
```

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

3)
```
   00001111
or 00010101
   00011111 = 31
```

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

5)
```
while (tp->next->next != NULL)
  tp = tp->next;
free(tp->next);
tp->next = NULL;
tp = lista;
```

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

void las_rtal(char *prompt, rtal *rp)
{
  if (prompt) printf(prompt);
  scanf("%d/%d", &rp->t, &rp->n);
  normera_rtal(rp);
}

void skriv_rtal(char *prompt, rtal r)
{
  if (prompt) printf(prompt);
  printf("%d/%d", r.t, r.n);
}
```

7)
```
void normera_rtal(rtal *rp)
{
  int sgd;

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

  if (rp->n < 0) {
    rp->t = -rp->t;
    rp->n = -rp->n;
  }
}
```

8)
```
int is_palindrom(char *str, size_t n)
{
  if (n < 2)
    return 1;
  else if (str[0] != str[n-1])
    return 0;
  else
    return is_palindrom(&str[1], n-2);
}
```

9)
```
unsigned char swap_nibbles(unsigned char uch)
{
  return ((uch << 4) | (uch >> 4));
}
```

Anm: Så länge man högerskiftar variabler av typen ”unsigned” används logiskt skift. Om variabeln däremot är av typen ”signed” är valet av logiskt eller aritmetiskt skift odefinierat i standarden och därmed implementationsberoende. För säkerhets skull bör man därför maska bort de bitar som skiftats in: `((sch >> 4) & 0xF)`

10)
```
int hashfunk_str(char *str)
{
  int i, sum = 0;

  for (i = 0; str[i] != '\0'; i++)
    sum += str[i];

  return sum;
}
```

(65 + 66 + 66 + 65) % 97 = 262 % 97 = 68

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

void main()
{
  int i, nr;
  rtal *rp, rsum, rnr;

  printf("Ge antal ratinella tal: ");
  scanf("%d", &nr);
  rp = calloc(nr, sizeof(rtal));
  skapa_rtal(&rsum, 0, 1);
  for (i = 0; i < nr; i++)
  {
    las_rtal("Ge rtal (t/n): ", &rp[i]);
    rsum = add_rtal(rsum, rp[i]);
  }
  skapa_rtal(&rnr, nr, 1);
  skriv_rtal("Medel = ", div_rtal(rsum, rnr));
  free(rp);
}
```

12)
```
rtal add_rtal(rtal r1, rtal r2)
{
  rtal sum;

  sum.n = lcm(r1.n, r2.n);
  sum.t = (sum.n/r1.n)*r1.t + (sum.n/r2.n)*r2.t;
  normera_rtal(&sum);
  return sum;
}

rtal sub_rtal(rtal r1, rtal r2)
{
  rtal diff;

  diff.n = lcm(r1.n, r2.n);
  diff.t = (diff.n/r1.n)*r1.t - (diff.n/r2.n)*r2.t;
  normera_rtal(&diff);
  return diff;
}

rtal mul_rtal(rtal r1, rtal r2)
{
  rtal prod;

  prod.t = r1.t * r2.t;
  prod.n = r1.n * r2.n;
  normera_rtal(&prod);
  return prod;
}

rtal div_rtal(rtal r1, rtal r2)
{
  rtal kvot;

  kvot.t = r1.t * r2.n;
  kvot.n = r1.n * r2.t;
  normera_rtal(&kvot);
  return kvot;
}
```

13)
```
#include <stdio.h>
#include "rtal.h"
#include "lifo.h"

void main()
{
  linktyp *lp = NULL;
  FILE *tsin;
  rtal key, r; int t, n;

  tsin = fopen("Rtal.txt", "rt");
  while (fscanf(tsin, "%d %d", &t, &n) != EOF) {
    skapa_rtal(&r, t, n); push(&lp, r);
  }
  fclose(tsin);
  las_rtal("Ge nyckel: ", &key);
  while (lp != NULL) {
    r = pop(&lp);
    if (mindre_rtal(key, r)) {
      skriv_rtal(NULL, r); skriv_rtal(" > ", key); putchar('\n');
    }
  }
}
```

14)
```
#include <stdio.h>
#include "dice.h"
#include "tree.h"

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

  init_dice(NULL);

  printf("Ge antalet kast: ");
  scanf("%d", &antal);
  for (i = 1; i <= antal; i++) {
    throw_dice(&d);
    into_tree(&rot, d, is_less_dice);
  }

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

  for (i = 1; i <= 6; i++) {
    put_dice(&key, i);
    printf("Antal %d:or = %d\n", i,
           same_in_tree(rot, key, is_equal_dice, is_less_dice));
  }
}
```

15)
```
#include <stdio.h>
#include "rtal.h"
#include "twolist.h"

void main()
{
  FILE  *bsin;
  headtyp *hp;
  linktyp *lp;
  rtal r, rkey;

  newhead(&hp);
  bsin = fopen("Rtal.dat", "rb");
  fread(&r, sizeof(rtal), 1, bsin);
  while(!feof(bsin)) {
    newlink(&lp);
    putlink(r, lp);
    insort(lp, hp, mindre_rtal);
    fread(&r, sizeof(rtal), 1, bsin);
  }
  fclose(bsin);

  skapa_rtal(&rkey, 0, 1);
  lp = firstlink(hp);
  while (lp != NULL) {
    r = getlink(lp);
    if (mindre_rtal(rkey, r))
      skriv_rtal("  ", r);
    lp = succlink(lp);
  }
  elimhead(&hp);
  putchar('\n');
}
```