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), SDU2, TDVB, 2006-10-31.

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.

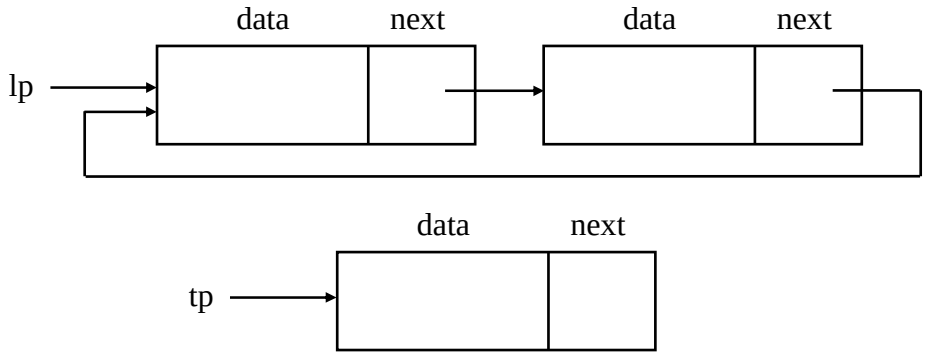
Lycka till!

---

1) Antag att du har en pekare **rp** som pekar på ett minnesutrymme som innehåller ett reellt tal. Skriv den sats som byter ut det reella talet mot kvadratroten av talet. **(1p)**

---

2) Stoppa in länken **tp** efter länken **lp** i den länkade strukturen nedan. Gör därefter **tp** till den första länken. Inga extra pekare får definieras och **tp** ska när allt är klart 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 = 0x5a;
```

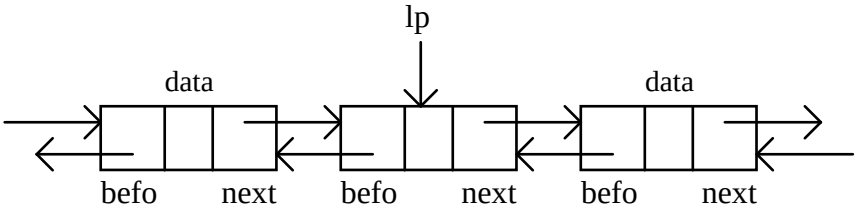
Ange värdet decimalt av **uch** efter satsen:

```
uch = uch & (1<<3);
```

---

4) Skriv de satser som krävs för att ta bort och avallokera länken som **lp** pekar på i nedanstående tvåvägslista. Inga färdiga rutiner eller extra pekare får användas och listan ska hänga ihop efteråt.

(1p)



---

5) Rita det binära träd som skapas då bokstäverna i ordet "programmera" instoppas bokstav för bokstav i trädet i den angivna ordningen, om instoppningsfunktionen sätter in data som är mindre i vänsterträdet och lika stora eller större i högerträdet. **(1p)**

---

6) En tidpunkt under ett dygn kan avbildas som en abstrakt datatyp enligt:

```
/* Specifikation av dygnstid -- tid.h     */
/* Vi normerar tiden så att h, m och s är */
/* >= 0 och att 00:00:00 <= t < 23:59:59  */


#ifndef TID_H
#define TID_H

typedef struct {
  int h;          /* Timmar:   0-23 */
  int m;          /* Minuter:  0-59 */
  int s;          /* Sekunder: 0-59 */
} tid;

void skapa_tid(tid *tp, int h, int m, int s);
/* Skapar och normerar en tid (se nedan). */

void las_tid(char *ledtext, tid *tp);
/* Skriver ut eventuell ledtext och läser därefter in en */
/* tid på formen hh:mm:ss från tangentbordet.            */
/* Efter inläsning normeras tiden (se nedan).            */

void skriv_tid(char *ledtext, tid t);
/* Skriver ut eventuell ledtext och en tid på skärmen. */
/* Utskriften är på formen hh:mm:ss.                   */

void normera_tid(tid *tp);
/* Normerar tiden så att 00:00:00 <= t < 23:59:59.  */

int sek_tid(tid t);
/* Returnerar tiden i antal sekunder 0 <= s < 24*60*60. */

void andra_tid(tid *tp, int s);
/* Ändrar tiden s sekunder och normerar resultatet */

tid add_tid(tid t1, tid t2);
/* Addition (t1 + t2) med normering.  */
/* Fungerar bäst om t1+t2 <= 23:59:59 */

tid sub_tid(tid t1, tid t2);
/* Subtraktion (t1 – t2) med normering. */
/* Fungerar bäst om t1 >= t2            */

int innan_tid(tid t1, tid t2);
/* Sant om dygnstiden t1 är före t2, annars falskt */

int samma_tid(tid t1, tid t2);
/* Sant om dygnstiden t1 är samma som t2, annars falskt */

#endif
```

Implementera funktionerna **las_tid** och **skriv_tid** enl. ovan. **(2p)**

---

7) Implementera funktionen **normera_tid** i uppgift 6 ovan. Efter normeringen skall tiden ligga mellan 00:00:00 och 23:59:59, så att 0:0:125 normeras till 00:02:05, 23:59:65 normeras till 00:00:05 och -1:1:-1 normeras till 23:00:59. Använd gärna **sek_tid**. **(2p)**

---

8) Skriv en rekursiv funktion som kopierar strängen **str2** till **str1**. Inga färdiga funktioner får användas. Funktionshuvud enligt: **(2p)**

```
void strkopia(char *str1, char *str2)
```

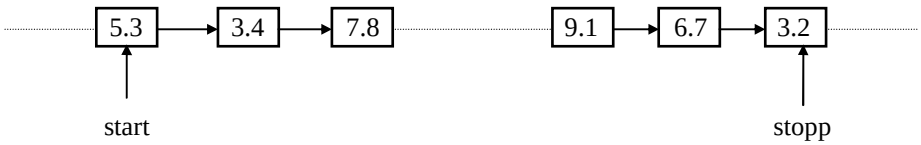
---

9) Fullborda funktionen **flest_ettor**, som ska returnera sant om antalet ettställda bitar i ASCII-koden för **ch** är fler än 4. Är exempelvis **ch** 'A' med ASCII-koden 01000001 ska funktionen returnera 0 och är **ch** 'g' med ASCII-koden 01100111 ska funktionen returnera 1. **(2p)**

```
int flest_ettor(char ch)
```

---

10) 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 **ta_bort_lankar** nedan, som tar **start** och **stopp** som parametrar och som tar bort och avallokerar alla länkar i listan mellan **start** och **stopp**. Start- och stopplänken ska finnas kvar i listan. **(2p)**

```
void ta_bort_lankar(linktyp *start, linktyp *stopp)
```

---

11) Implementera funktionerna **skapa_tid**, **sek_tid** och **andra_tid** enligt uppgift 6 ovan. Skriv därefter ett huvudprogram som testar funktionerna genom att skapa två tider och därefter subtraherar den ena tiden från den andra. **(5p)**

---

12) Skriv ett program som dynamiskt allokerar en vektor som ska innehålla ett inläst antal tider enligt uppgift 6 ovan. Läs in tiderna till vektorn och skriv därefter ut medelvärdet av tiderna i vektorn. Observera att även medelvärdet ska vara en tid. Använd därför **sek_tid** och **skapa_tid** när medelvärdet beräknas. **(5p)**

---

13) Skriv ett fullständigt program som läser en källkodsfil i C och kontrollerar att alla parenteser och klamrar (som ( ), { } och [ ]) är matchade. Problemet ska lösas genom att när vänsterparenteser och klamrar läses från filen läggs de på en stack i väntan på att motsvarande högerparentes eller klammer läses. När exv. en högerparentes läses så poppas översta tecknet från stacken (om den inte är tom). Detta tecken ska vara motsvarande vänsterparentes. Om så ej är fallet (eller om stacken är tom) har matchningen misslyckats. När filen är slut ska också stacken vara tom. För enkelhets skull kan du bortse från att parentestecken och klamrar kan förekomma omatchade i kommentarer, strängar etc. Använd funktionerna **is_left**, **is_right** och **is_match** samt **push** och **pop** för att lösa problemet. (Se nedan.) **(5p)**

För tester på parenteser och klamrar ska du använda:

```
int is_left(int ch);
/* Returnerar 1 (sant) om ch är en vänsterparentes */
/* eller vänsterklammer etc., annars 0 (falskt)    */

int is_right(int ch);
/* Returnerar 1 (sant) om ch är en högerparentes */
/* eller högerklammer etc., annars 0 (falskt)    */

int is_match(int left, int right);
/* Returnerar 1 (sant) om left och right   */
/* är ett matchande par, annars 0 (falskt) */
```

För hantering av stacken ska du använda:

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

#ifndef LIFOH
#define LIFOH

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 */

#endif
```

---

14) Textfilen **Tider.txt** innehåller ett antal tider på formen hh:mm:ss enligt uppgift 6 ovan. Skriv ett program som läser alla tider från textfilen och placerar dessa 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 och för att hantera tider ska du endast använda den abstrakta datatypen **tid**. Tråkigt nog stämmer inte parameterlistan för **skriv_tid** i **tid** överens med parameterlistan för **write_data** i **show_tree** i **tree**, så detta måste fixas på något sätt. Textfilen ser ut så här: **(5p)**

```
11:22:33
14:12:10
00:00:00
23:59:59
12:00:00
11:59:59
05:05:05
22:22:22
  ....
```

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

#ifndef TREE_H
#define TREE_H

#include "tid.h"
typedef tid 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) I binärfilen **Tider.dat** finns ett antal tidpunkter av den abstrakta datatypen **tid**, enligt uppgift 6 ovan. Skriv ett fullständigt program som läser alla tidpunkter från filen och stoppar in dessa i en tvåvägslista sorterad enligt innan-funktionen. Avslutningsvis ska programmet skriva ut de sorterade tiderna radvis på skärmen med rubriker för förmiddagstider resp. eftermiddagstider enligt: **(5p)**

```
Förmiddag
00:00:00
05:05:05
11:22:33
11:59:59

Eftermiddag
12:00:00
14:12:10
22:22:22
23:59:59
```

För hantering av tider ska du endast använda den abstrakta datatypen **tid** i uppgift 6 ovan och för hantering av tvåvägslistan ska du endast använda den abstrakta datatypen **twolist** nedan.

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

#ifndef TWOLISTH
#define TWOLISTH

#include "tid.h"
typedef tid 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 2006-10-31

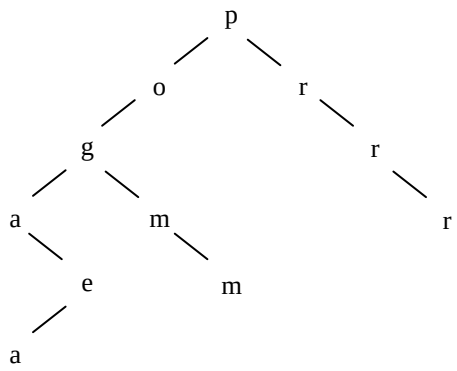
1)
```
*rp = sqrt(*rp);
```

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

3)
```
  01011010
& 00001000
  00001000 = 8
```

4)
```
lp->befo->next = lp->next;
lp->next->befo = lp->befo;
free(lp);
lp = NULL;
```

5)



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

void las_tid(char *ledtext, tid *tp)
{
  if (ledtext) printf(ledtext);
  scanf("%d:%d:%d", &tp->h, &tp->m, &tp->s);
  normera_tid(tp);
}


void skriv_tid(char *ledtext, tid t)
{
  if (ledtext) printf(ledtext);
  printf("%02d:%02d:%02d", t.h, t.m, t.s);
}
```

7)
```
void normera_tid(tid *tp)
{
  int sek = sek_tid(*tp);

  tp->s = sek%60;
  tp->m =(sek/60)%60;
  tp->h =(sek/(60*60))%24;
}
```

8)
```c
void strkopia(char *str1, char *str2)
{
  *str1 = *str2;
  if (*str2 != '\0')
    strkopia(str1+1, str2+1);
}
```

9)
```c
int flest_ettor(char ch)
{
  int i, sum = 0;

  for (i = 0; i <= 7; i++)
    if (ch & (1 << i)) sum++;
  return sum > 4;
}
```

10)
```c
void ta_bort_lankar(linktyp *start, linktyp *stopp)
{
  linktyp *temp;

  temp = start->next;
  while (temp != stopp) {
    start->next = temp->next;
    free(temp);
    temp = start->next;
  }
}
```

11)
```c
void skapa_tid(tid *tp, int h, int m, int s)
{
  tp->h = h; tp->m = m; tp->s = s;
  normera_tid(tp);
}

int sek_tid(tid t)
{
  int sek = (t.h*60*60 + 60*t.m + t.s) % (24*60*60);

  if (sek < 0)
    sek = (sek + 24*60*60);
  return sek;
}

void andra_tid(tid *tp, int s)
{
  tp->s += s;
  normera_tid(tp);
}

#include <stdio.h>
#include "tid.h"

void main(void)
{
  tid t1, t2;

  skapa_tid(&t1, 12,  0, 30);
  skapa_tid(&t2,  6, 30, 45);
  skriv_tid(NULL, t1); skriv_tid(" - ", t2);
  andra_tid(&t1, -1*sek_tid(t2));
  skriv_tid(" = ", t1); putchar('\n');
}
```

12)

```c
#include <stdio.h>
#include <stdlib.h>
#include "tid.h"

void main()
{
  int i, nr, sum = 0;
  tid *tp, medel;

  printf("Ge antal tider: ");
  scanf("%d", &nr);
  tp = calloc(nr, sizeof(tid));
  for (i = 0; i < nr; i++) {
    las_tid("Ge tid (hh:mm:ss): ", &tp[i]);
    sum += sek_tid(tp[i]);
  }
  skapa_tid(&medel, 0, 0, sum/nr);
  skriv_tid("Medel = ", medel); putchar('\n');
  free(tp);
}
```

13)

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

int main()
{
  linktyp *lp = NULL;
  FILE *tsin;
  int ch;

  tsin = fopen("Fil.c", "rt");
  while ((ch = fgetc(tsin)) != EOF) {
    if (is_left(ch))
      push(&lp, ch);
    else if (is_right(ch)) {
      if (lp == NULL || !is_match(pop(&lp), ch)) {
        printf("Omatchad %c hittad!\n", ch);
        exit(2);
      }
    }
  }
  fclose(tsin);

  if (lp != NULL) {
    printf("Omatchad %c hittad!\n", pop(&lp));
    exit(1);
  }
  else {
    printf("Filen är ok.\n");
    exit(0);
  }
}
```

14)

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

void write_data(tid t)
{
  skriv_tid("\n", t);
}
```

```c
void main()
{
  FILE  *tsin;
  nodtyp *rot = NULL;
  int h, m, s;
  tid t, key;

  tsin = fopen("Tider.txt", "rt");
  while (fscanf(tsin, "%2d:%2d:%2d", &h, &m, &s) != EOF) {
    skapa_tid(&t, h, m, s);
    into_tree(&rot, t, innan_tid);
  }
  fclose(tsin);

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

  las_tid("Ge nyckel: (hh:mm:ss): ", &key);
  if (search_tree(rot, key, samma_tid, innan_tid))
    printf("Nyckeln finns i tädet!\n");
  else
    printf("Nyckeln finns ej i tädet!\n");
}
```

15)

```c
#include <stdio.h>
#include "tid.h"
#include "twolist.h"

void main()
{
   FILE  *bsin;
   headtyp *hp;
   linktyp *lp;
   tid t, key;
   int fm = 1;

   newhead(&hp);
   bsin = fopen("Tider.dat", "rb");
   fread(&t, sizeof(tid), 1, bsin);
   while(!feof(bsin)) {
     newlink(&lp);
     putlink(t, lp);
     insort(lp, hp, innan_tid);
     fread(&t, sizeof(tid), 1, bsin);
   }
   fclose(bsin);

   skapa_tid(&key, 12, 0, 0);
   lp = firstlink(hp);
   printf("\nFörmiddag");
   while (lp != NULL) {
     t = getlink(lp);
     if (fm && !innan_tid(t, key)) {
       printf("\n\nEftermiddag");
       fm = 0;
     }
     skriv_tid("\n", t);
     lp = succlink(lp);
   }
   elimhead(&hp);
   putchar('\n');
}
```