ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lösningarna till tentamensuppgifterna sätts ut på kurssidan på nätet i dag kl 19.

Denna tenta kommer att vara färdigrättad Fr 3/9 -04 och kan då hämtas på mitt tjänsterum T2221 mellan 13 och 15. Vid detta tillfälle har du också möjlighet att diskutera rättningen. Tentor som ej hämtats då placeras på studentexpedition.

Omtentamen i Programmeringsmetodik, 5p, Au2, D1, E1, Pr1, 040827.

Hjälpmedel : Inga
Tid : 14-19
Ansvarig lärare : Gunnar Joki 303317, 274825(hem)

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 efter uppgiftens nummer. 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 Borland C.

Detta häfte ska du behålla.

Lycka till!

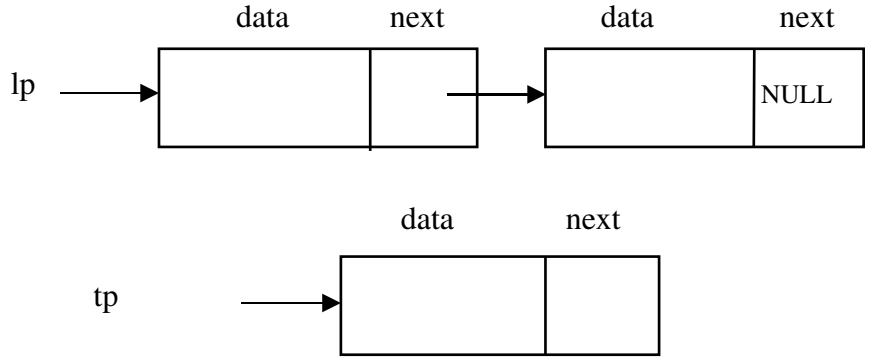
1) (1p) Anta att du har en pekare given enligt:

```
float *fv;
```

Visa hur du använder denna pekare till att allokera en vektor för 100 reella tal.

---

2)(1p) Stoppa in länken tp mitt emellan de två länkarna i den länkade strukturen nedan. Inga extra pekare får definieras.



---

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

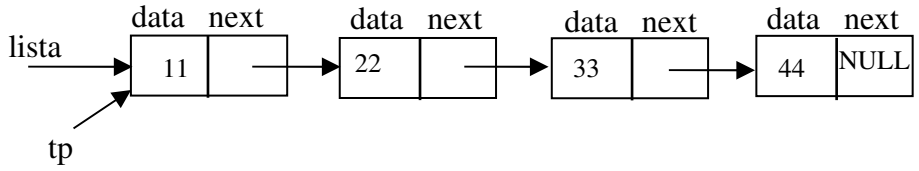
```
unsigned char uch = 0xff;
```

Ange värdet decimalt av uch efter satsen:

```
uch &= 45;
```

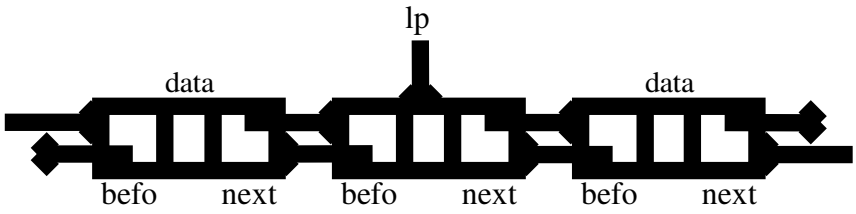
---

4) (1p) Skriv de satser som tar bort och avallokerar länken med värdet 33 nedan. Efter bortplockandet ska lista och tp peka på den första länken precis som innan och hela listan ska hänga ihop.



---

5)(1p) Ange hur du genom att använda de färdiga funktionerna för tvåvägslistor(se uppgift 15) skapar en ny länk tp med samma datadel som lp och sätter in den efter lp i listan.



6)(2p) En student kan avbildas som en abstrakt datatyp med personnummer och avklarade studiepoäng enligt:

```
/* Student.h */

typedef
struct
{
   char pnr[12];  /* Personnummer */
   float poang;   /* Studiepoäng */
} student;

void las_student(student *sp);
/* Läser in pnr och poang med ledtexter*/

void skriv_student(student s);
/* Skriver ut pnr och poang med ledtexter */

int flick_student(student s);
/* Returnerar sant (1) om studenten är flicka annars falskt (0)*/

int jfr_studenter(student s1, student s2));
/* Returnerar sant (1) om s1 har mindre studiepoäng än s2 */

int samma_student(student s1, student s2));
/* Returnerar sant (1) om s1 har samma personnummer som s2 */
```

Implementera funktionen las_student.

---

7)(2p) Implementera funktionen skriv_student, i uppgift 6 ovan.

---

8)(2p) Skriv ett huvudprogram som skapar två studenter, enligt uppgift 6 ovan, läser in värden till dessa och skriver ut dem i ordning med den student som har flest studiepoäng först. Använd funktionerna för den abstrakta datatypen student.

---

9)(2p) Skriv en funktion som tar en pekare till ett heltal och ett bitnummer (0 – 31) som parametrar och som nollställer den angivna biten. Inga andra bitar får påverkas. Funktionsprototyp enligt:

```
void nolla_bit(int *talpek, int bit_nr);
```

---

10)(2p)Antag att du i en lista enligt uppgift 4 ovan har data i form av studenter enligt uppgift 6 ovan. Skriv en rekursiv funktion, som letar efter en given student. Finns studenten i listan, returnerar funktionen sant (1) annars falskt (0). Använd funktionerna för den abstrakta datatypen student. Funktionshuvud enligt:

```
int reksok(linktyp *lista, student key);
```

11)(5p)Skriv ett fullständigt program som läser in två reella tal, byter värden mellan talen och skriver ut de nya värdena. Bytet ska göras av en funktion med prototypen :

```
void byt(float *ap, float *bp);
```

---

12)(5p)Skriv ett fullständigt program som upprepat (avslut med 0) läser in aktuellt antal element i en heltalsvektor, allokerar minne dynamiskt för denna vektor och fyller vektorn med slumpade tresiffriga tal mellan 100 och 999. Efter slumpning ska programmet skriva ut vektorn baklänges med 10 tal per rad förutom sista raden där antalet tal varierar beroende på det totala antalet. Glöm ej att avallokera dynamiskt minne så att programmet ej äter minne.

---

13)(5p)Fullborda nedanstående program genom att i main-programmet skriva koden som utför

det som står i kommentarerna. Programmet använder en hashtabell med öppen adressering för kollisioner för att kontrollera om en student läser en viss kurs. Du ska anropa de givna funktionerna :

```
#include ”Student.h”

void into_openhash(int *hashtab[], student s);
/* Stoppar in studenten i hashtab */

int search_openhash(int *hashtab[], student key_student);
/* Om det finns en student med key-studentens personnummer i tabellen
returneras det index som studenten finns på annars returneras -1 */


void main()
{

   student sv[5] = {{”711111-1111”, 43}, {”821212-2222”, 35},……}};
   student key_student;
   int *hashtabell[10], i, keyindex;

   /* Sätt hashtabellens alla element till NULL */
   /* Stoppa in alla studenter från vektorn sv i hashtabellen */
   /* Läs in key_student */
```

```
   /* Sök i hashtabellen efter key_student */
   /* Skriv sökresultat */
}
```

14)(5p)I textfilen Student.txt finns ett antal studenter med personnummer och studiepoäng enligt:

```
831203-3323       21.5
820912-2345       16.0
820415-1236       80.5
. . . . . .
```

Skriv ett fullständigt program som läser alla studenter från filen och stoppar in dessa på en stack. Töm sedan stacken och skriv ut alla studenter som har fler studiepoäng än ett inläst värde. För hantering av stacken ska du använda:

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

#include ”Student.h”
typedef student datatyp;

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 */
```

---

15)(5p)Binärfilen Student.dat innehåller ett antal studenter enligt uppgift 6 ovan. Skriv ett program som läser alla studenter från binärfilen och placerar alla flickor i början av en tvåvägslista och alla pojkar i slutet av listan. Skriv sedan tillbaks studenterna till samma fil som nu får en uppdelning i flickor och pojkar. Du ska även skriva funktionen flickstudent enligt uppgift 6 ovan som du anropar för att skilja på flickor och pojkar. Flickorna har jämn näst sista siffra i personnumret, medan pojkarna har udda. För hantering av tvåvägslistan ska du använda:

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

```
#include ”Student.h”
typedef student 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 tentamen i Programmeringsmetodik, 5p, 040827

1)
```
fp = calloc(100, sizeof(float));
```

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

3)
```
45
```

4)
```
tp = tp->next->next;
lista->next->next = tp->next;
free(tp);
```

5)
```
newlink(&tp);
putlink(getlink(lp), tp);
insucc(tp, lp);
```

6)
```
#include ”Student.h”
#include <stdio.h>

void las_student(student *sp)
{
   printf("Ge personnummmer : ");
   scanf("%s", sp->pnr);
   printf("Ge studiepoäng : ");
   scanf("%f", &sp->poang);
}
```

7)
```
#include "Student.h"
#include <stdio.h>

void skriv_student(student s)
{
   printf(”Personnummer : %s\n”, s.pnr);
   printf(”Studiepoäng : %.1f\n”, s.poang);
}
```

8)
```
#include ”Student.h”
#include <conio.h>

void main()
{
   student a, b;

   las_student(&a);
   las_student(&b);
   if (jfr_studenter(a, b))
   {
      skriv_student(b);
      skriv_student(a);
   }
   else
   {
      skriv_student(a);
      skriv_student(b);
   }
   getch();
}
```

9)
```
void nolla_bit(int *talpek, int bit_nr)
{
    *talpek = *talpek & (~(1 << bit_nr));
}
```

10)
```
int rek_sok(linktyp *lista, student key)
{
   if (lista == NULL)
      return 0;
   else if (samma_student(lista->data, key))
      return 1;
   else
      return rek_sok(lp->next, key);
}
```

11)
```
#include <stdio.h>
#include <conio.h>

void byt(float *ap, float *bp)
{
   float akopia = *ap;
   *ap = *bp;
   *bp = akopia;
}

void main()
{
   float a, b;

   printf(”Ge ett reellt tal a : )
   scanf(”%f”, &a);
   printf(”Ge ett reellt tal b : )
   scanf(”%f”, &b);
   byt(&a, &b);
   printf(”Talet a är efter bytet : %f\n”, a);
   printf(”Talet b är efter bytet : %f\n”, b);
   getch();
}
```

12)
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void main()
{
   int *iv, i, antal;

   srand((unsigned)time(NULL));

   printf("Ge antalet element i vektorerna (avslut 0) : ");
   scanf("%d", &antal);
   while (antal > 0)
   {
      iv[i] = calloc(antal, sizeof(int));
      for (i = 0; i < antal; i++)
{
         iv[i] = rand() % 900 + 100;
      }
      for (i = antal - 1; i >= 0; i--)
      {
        printf(”%4d”,iv[i]);
        if ((antal - i) % 10 == 0)
```

```
            printf(”\n”);
         }
         free(iv);
         printf("Ge antalet element i vektorerna (avslut 0) : ");
         scanf( "%d", &antal);
      }
   }
```

13)
```
#include <stdio.h>
#include <conio.h>

void main()
{

   student sv[5] = {{”711111-1111”, 43}, {”821212-2222”, 35},……}};
   student key_student;
   int *hashtabell[10], i, keyindex;

   /* Sätt hashtabellens alla element till NULL */
   for (i=0; i < 10; i++)
      hashtabell[i]= NULL;

   /* Stoppa in alla studenter från vektorn sv i hashtabellen */
   for (i=0; i < 5; i++)
      into_openhash(hashtabell, sv[i]);

   /* Läs in key_student */
   printf(”Vem söks?\n”);
   las_student(&key_student);

   /* Sök i hashtabellen efter key_student */
   keyindex = search_openhash(hashtabell, key_student);

   /* Skriv sökresultat */
   if (keyindex == -1)
      printf(”Studenten finns ej!”);
   else
      printf(”Studenten finns på index %d”, i);
   getch();
}
```

---

14)
```
#include <stdio.h>
#include <conio.h>
#include "Lifo.h"

void main()
{
   FILE *tsin;
   linktyp *lp = NULL;
   student s;
   float minpoang;

   tsin = fopen(”Student.txt”,”rt”);
   while (fscanf(tsin, ”%s%f”, s.pnr, &s.poang) != EOF)
   {
      push(&lp, s);
   }
   fclose(tsin);
   printf(”Ge minpoang : ”);
   scanf(”%f”, &minpoang);
   while (lp != NULL)
```

```
   {
      s = pop(&lp);
      if (s.poang >= minpoang)
      {
         skriv_student(s);
      }
   }
   getch();
}
```

15)

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

int flick_student(student s)
{
   return s.pnr[9] % 2 == 0;

}

void main()
{
   FILE *bsinut;
   headtyp *hp;
   linktyp *lp;
   student s;

   newhead(&hp);
   bsinut = fopen("Student.dat", "r+b");
   fread(&s, sizeof(student), 1, bsinut);
   while(!feof(bsin))
   {
      newlink(&lp);
      putlink(s, lp);
      if(flick_student(s))
         infirst(lp, hp);
      else
         inlast(lp, hp);
      fread(&s, sizeof(tvapol), 1, bsinut);
   }
   rewind(bsinut);
   lp = firstlink(hp);
   while (lp != NULL)
   {
      s = getlink(lp);
      fwrite(&s, sizeof(student), 1, bsinut);
      lp = succlink(lp);
   }
   fclose(bsinut);
   elimhead(&hp);
   getch();
}
```