



ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lämna ifylld kursvärdering till skrivvakten samtidigt med tentan.

Lösningarna till tentamensuppgifterna sätts ut på kurssidan på nätet idag kl 19.

Denna tenta kommer att vara färdigrättad On 11/6 och kan då hämtas på mitt tjänsterum T2221 mellan kl 10 och 12. Vid detta tillfälle har du också möjlighet att diskutera rättningen. Tentor som ej hämtats då placeras på studentexpedition.

Tentamen i Programmeringsmetodik, 5p, Au2, D1, E1, Pr1, 030603.

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.

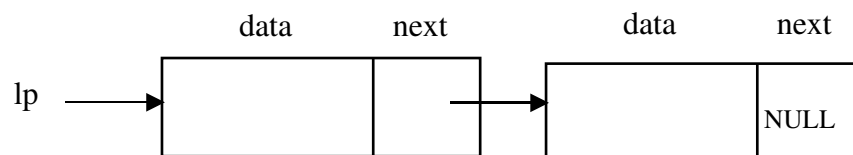
Om inget speciellt anges gäller frågorna Borland C.

Detta häfte ska du behålla.

Lycka till!

1) (1p) Antag att du har en pekare `ap` som pekar på ett minnesutrymme som innehåller ett reellt tal. Skriv den sats som halverar detta tal.

2)(1p) Skriv den sats som beräknar medelvärdet av de två länkade structarnas data, i nedanstående figur. Data är ett reellt tal av typen float. Du får inte definiera några nya pekare.



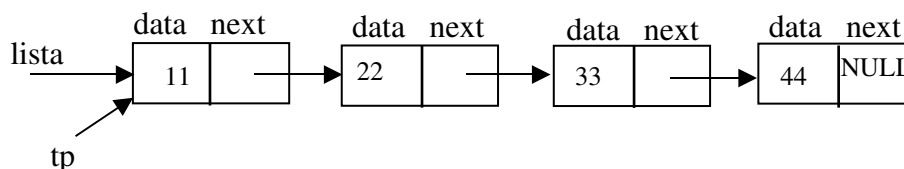
3) (1p) Antag att du har en 8 bitars unsigned char definierad enligt:

```
unsigned char uch = 5;
```

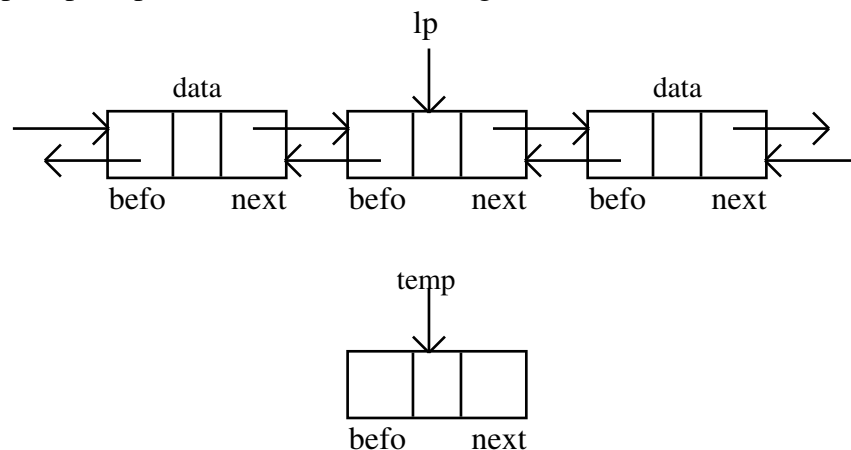
Ange värdet av `uch` efter satsen:

```
uch |= 2;
```

4) (1p) Skriv de satser som tar bort och avallokerar den första (längst till vänster) länken i en lista av nedanstående typ.



5) (1p) Ange hur du med hjälp av funktionerna för tvåvägslistor (se uppgift 15) stoppar in länken `temp` före (till vänster om) länken `lp` i nedanstående tvåvägslista och därefter sätter `lp` att peka på listans sista länk. Tvåvägslistans huvud har adressen `minlista`.



- 6)(2p) En geometrisk talföljd som ex. 1.5, 0.75, 0.375,, kan avbildas som en abstrakt datatyp bestående av första talet a och den konstanta kvoten k mellan ett tal och närmast föregående tal enligt:

```

/* Getal.h */

typedef
struct
{
    float a;    /* Första talet ex ovan 1.5 */
    float k;    /* Kvoten ex ovan 0.75/1.5 = 0.5 */
} getal;

void las_getal(getal *gp);
/* Läser in a och k för talföljden */

float get_tal(getal g, int n);
/* Returnerar tal  $ak^n$  i följderna */

void skriv_getal(getal g, int nr);
/* Skriver ut de första nr talen ex 1.5, 0.75, 0.375 om nr är 3 */

int is_konvergent_getal(getal g);
/* Sant om  $-1 < k < 1$  annars falskt */

float summa_getal(getal g);
/* Summan av alla tal i en konvergent serie är  $a/(1-k)$  */

```

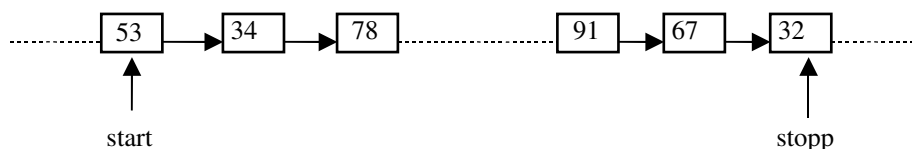
Implementera funktionen `las_getal`.

- 7)(2p) Implementera funktionen `get_tal` ovan.
-

- 8)(2p) Skriv funktionen `nollor` så att den returnerar antalet nollställda bitar i heltalet tal som består av 32 bitar. Funktionsprototyp enligt:

```
int nollor(int tal);
```

- 9) (2p) Antag att du har en envägslista, där länkarna är av samma typ, `linktyp`, som i uppgift 2 ovan men med data i form av hela tal, enligt:



Skiv funktionen `modify` nedan, så att den adderar heltalsparametern `tal` till varje länks data från och med `start` till och med `stopp`. Funktionsprototyp enligt:

```
void modify(linktyp *start, linktyp *stopp, int tal);
```

10)(2p) Skriv en rekursiv funktion som åstadkommer samma sak som funktionen modify i uppgift 9 ovan. Funktionshuvud enligt:

```
void rek_modify(linktyp *start, linktyp *stopp, int tal);
```

11)(5p) Skriv ett fullständigt program som upprepat (avslut 0) frågar efter hur många tresiffriga tal som ska slumpas till en vektor. Vektorn ska allokeras dynamiskt så att talen precis ryms i den. Efter slumpningen ska vektorn skrivas ut, sorteras, skrivas ut på nytt och avallokeras. För utskrift och sortering ska du anropa funktionerna skriv_vek och sort_vek, som du får anta givna. De givna funktionerna har prototyper enligt:

```
void skriv_vek(int v[], int nr);
/* Skriver ut vektorn v som har nr element*/

void sort_vek(int v[], int nr);
/* Sorterar vektorn v som har nr element */
```

12)(5p) Fullborda huvudprogrammet nedan så att den stoppar in registreringsnumren från den givna vektorn i en hashtabell med öppen adressering och sedan frågar efter vilket registreringsnummer som söks och skriver ut om detta nummer finns i tabellen eller ej. Du ska anropa de givna funktionerna into_openhash och is_in_openhash med prototyper enligt:

```
void into_openhash(char *htab[], char *regnr);
/* Stoppar in registreringsnumret regnr i hashtabellen htab */

int is_in_openhash(char *htab[], char *keyregnr);
/* Returnerar sant(1) om keyregnr finns i htab annars falskt(0)*/

void main()
{
    char *vek[5] = {"HET657", "DRE123", "JYT786", "KLI877", "GTR697"};
    char *hashtab[10], sokt_regnr[7];
    int i, finns;

    /* NULL-ställ hashtab */

    /* Stoppa in alla registreringsnummer från vek in i hashtab */

    /* Läs in sökt registreringsnummer */

    /* Sök i hashtab */

    /* Skriv sökresultat */

}
```

13)(5p) Implementera funktionerna `skriv_getal`, `is_konvergent_getal` och `summa_getal`, enligt uppgift 6 ovan och skriv ett huvudprogram som läser in en geometrisk talföljd, skriver ut de 5 första talen i följd och om `getal` är konvergent beräknar och skriver summan.

14)(5p) Skriv ett fullständigt program som läser alla reella tal från textfilen `Retal.txt` och lägger dessa på en stack. Talen är alla positiva och åtskilda med blanktecken eller ny rad. Programmet ska sedan tömma stacken och skriva ut talen korrigerade, genom att subtrahera alla tal med det sist instoppade talet. Det tal som lästes sist ska alltså korrigerat bli 0.0 och skrivas ut först. För hantering av stacken ska du använda:

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

15)(5p) Skriv ett fullständigt program som läser en geometrisk talföljd med funktionen `las_getal`, enligt uppgift 6 ovan, räknar ut värdet för ett inläst antal tal i denna talföljd och stoppar in de uträknade talen i en tvåvägslista. Programmet ska sedan fortsätta med att upprepat (avslut 0) fråga efter hur många tal i listan som ska ingå i en delsumma som beräknas och jämförs med summan som beräknas med funktionen `summa_getal`. Båda summorna ska skrivas ut på skärmen. För hantering av tvåvägslistan ska du använda:

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

typedef float 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);
/* Elimineras och NULL-ställer listan */
```

Lösningar till tentamen i Programmeringsmetodik, 5p, 030603

- 1) `*ap = *ap/2;`
- 2) `medel = (lp->data + lp->next->data) / 2;`
- 3) `7`
- 4) `lista = lista->next;
free(tp);
tp = lista;`
- 5) `inpred(temp, lp);
lp = lastlink(minlista);`
- 6) `#include "Getal.h"
#include <stdio.h>

void las_getal(getal *gp)
{
 printf("Första talet ? ");
 scanf("%f", &gp->a);
 printf("Kvoten ? ");
 scanf("%f", &gp->k);
}`
- 7) `#include "Getal.h"
#include <math.h>

float get_tal(getal g, int n)
{
 return g.a * pow(g.k, n);
}`
- 8) `int nollor(int tal)
{
 int summa = 0, i;

 for (i = 0; i <= 31; i++)
 {
 if ((tal & (1 << i)) == 0)
 {
 summa++;
 }
 }
 return summa;
}`
- 9) `void modify(linktyp *start, linktyp *stopp, int tal)
{
 while (start != stopp)
 {
 start->data += tal;
 start = start->next;
 }
 stopp->data += tal;
}`

```

10) void rek_modify(linktyp *start, linktyp *stopp, int tal)
    {
        if (start != stopp)
            rek_modify(start->next, stopp, tal);
        start->data += tal;
    }

```

```

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

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

    printf("Ge antal tal som ska slumpas (avslut 0) : ");
    scanf("%d", &antal);
    srand((unsigned)time(NULL));
    while (antal > 0)
    {
        ivek = calloc(antal, sizeof(int));
        for ( i = 0; i < antal; i++)
        {
            ivek[i] = rand() % 900 + 100;
        }
        skriv_vek(ivek, antal);
        sort_vek(ivek, antal);
        skriv_vek(ivek, antal);
        free(ivek);
        printf("Ge antal tal som ska slumpas (avslut 0): ");
        scanf("%d", &antal);
    }
}

```

```

12) #include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void into_openhash(char *htab[], char *regnr);
/* Stoppar in registreringsnumret regnr i hashtabellen htab */

int is_in_openhash(char *htab[], char *keyregnr);
/* Returnerar sant(1) om keyregnr finns i htab annars falskt(0)*/

void main()
{
    char *vek[5] = {"HET657", "DRE123", "JYT786", "KLI877", "GTR697"};
    char *hashtab[10], sokt_regnr[7];
    int i, finns;

    /* NULL-ställ hashtab */
    for (i = 0; i < 10; i++)
    {
        hashtab[i] = NULL;
    }

    /* Stoppa in alla registreringsnummer från vek in i hashtab */

```



```

for (i = 0; i < 5; i++)
{
    into_openhash(hashtab, vek[i]);
}

/* Läs in sökt registreringsnummer */
printf("Ge sökt regnr : ");
gets(sokt_regnr);

/* Sök i hashtab */
finns = is_in_openhash(hashtab, sokt_regnr);

/* Skriv sökresultat */
if (finns)
    printf("Bilen finns i registret!");
else
    printf("Bilen finns ej i registret!");
}

```

13)

```

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

void skriv_getal(getal g, int nr)
{
    int i;

    for (i = 0; i < nr; i++)
    {
        printf("%f, ", get_tal(g, i));
    }
    printf("\n");
}

int is_konvergent_getal(getal g)
{
    return g.k > -1 && g.k < 1;
}

float summa_getal(getal g)
{
    return g.a / (1-g.k);
}

#include <conio.h>

void main()
{
    getal gt;

    las_getal(&gt);
    skriv_getal(gt, 5);
    if (is_konvergent_getal(gt))
        printf("Summa = %f", summa_getal(gt));
    getch();
}

```

14)

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "lifo.h"

```

```

void main()
{
    linktyp *lp = NULL;
    FILE *tsin;
    float x, sista;

    tsin = fopen("Retal.txt", "rt");
    while (fscanf(tsin, "%f", &x) != EOF)
    {
        push(&lp, x);
    }
    fclose(tsin);
    sista = x;
    while (lp != NULL)
    {
        printf("%.2f\n", pop(&lp) - sista);
    }
    getch();
}

```

15)

```

#include <stdio.h>
#include <conio.h>
#include "Twolist.h"
#include "Getal.h"
void main()
{
    headtyp *hp;
    linktyp *lp;
    getal gt;
    int i, tot_antal, del_antal;
    float tal, delsum;

    las_getal(&gt);

    printf("Hur många tal ska skapas ? ");
    scanf("%d", &tot_antal);

    newhead(&hp);
    for (i = 0; i < tot_antal; i++)
    {
        newlink(&lp);
        tal = get_tal(gt, i);
        putlink(tal, lp);
        inlast(lp, hp);
    }

    printf("Hur många tal i delsumman (avslut 0) ? ");
    scanf("%d", &del_antal);
    while (del_antal > 0)
    {
        i = 1;
        delsum = 0;
        lp = firstlink(hp);
        while (lp != NULL && i <= del_antal)
        {
            delsum += getlink(lp);
            lp = succlink(lp);
            i++;
        }
        printf("Delsumma = %f\n", delsum);
        printf("Totsumma = %f\n", summa_getal(gt));
    }
}

```

```
        printf("Hur många tal i delsumman (avslut 0) ? ");
        scanf("%d", &del_antal);
    }
    elimhead(&hp);
}
```