ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Kod : .....

Omtentamen i Programmeringsmetodik, 5p, Au2, D1 och E1, 020109

Hjälpmedel	: Inga	
Tid	: 8-13	
Ansvarig lärare	: Gunnar Joki	Tel arb: 303317
		Tel hem: 274825

Svaren till uppgifterna 1-15 ska skrivas på tillgängligt utrymme i detta häfte. Behöver du mera utrymme kan du skriva på baksidan eller på extra papper. Lösningarna till uppgifterna 16-19 ska skrivas på utdelat extra papper med maximalt en uppgift per papper. Skriv din kod på varje inlämnat extrapapper.

Den maximala poängen för respektive uppgift står angiven efter uppgiftens nummer. Totalt kan 40 poäng erhållas. För godkänt krävs ca 20 , för betyget 4 ca 28 och för betyget 5 ca 34 poäng.

Om inget speciellt anges gäller frågorna Borland C .

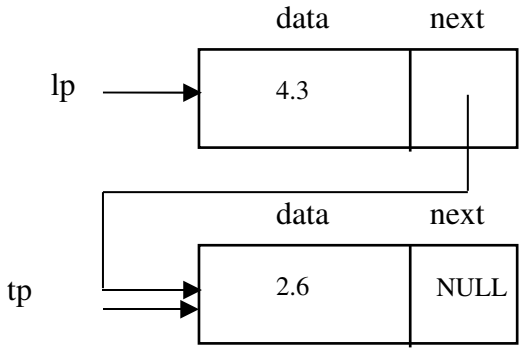
Detta häfte inlämnas.

Lycka till!

1) (1p) Antag att du har en pekare rp som pekar på ett minnesutrymme som innehåller ett reellt tal. Skriv den sats som fördubblar värdet av detta tal.

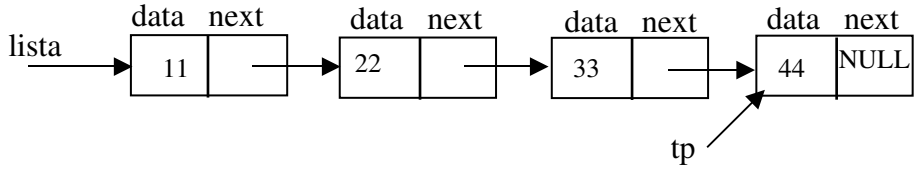
---

2)(1p) Skriv de satser som avallokerar den första (översta) länken och sätter lp att peka på den kvarvarande länken. Du får ej definiera några nya pekare.



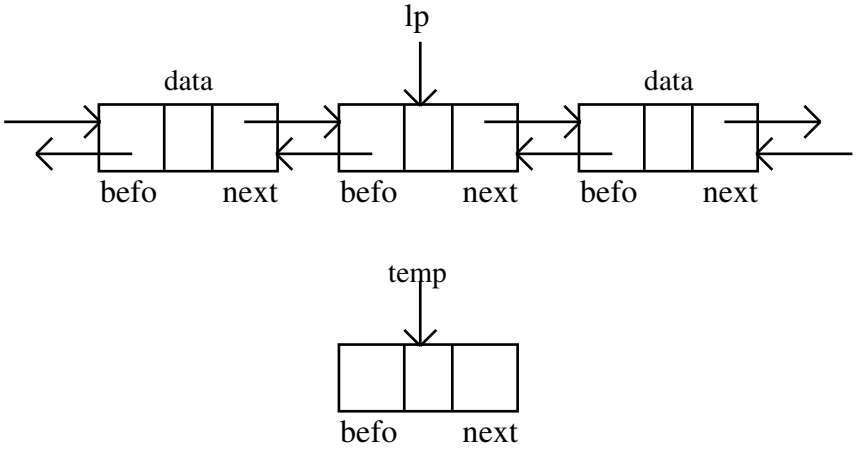
---

3) (1p) Antag att du vill ha in en ny länk med data lika med 55 sist (längst till höger) i nedanstående lista. Skriv de satser som dynamiskt skapar denna länk och stoppar in den i listan vars länkar är av linktyp. Efter instoppningen ska den nya listan avslutas med NULL och tp ska peka på sista länken. Inga extra pekare får användas.



---

4) (1p) Skriv de satser som krävs för att stoppa in länken som temp pekar före (till vänster om) länken som lp pekar på i nedanstående tvåvägslista. Inga färdiga rutiner eller extra pekare får användas och listan ska hänga ihop efteråt.



5) (1p) Rita det binära träd som skapas då talen 23, 16, 34, 13 och 14 instoppas tal för tal i trädet i den angivna ordningen, om instoppningsfunktionen sätter in data som är mindre i vänsterträdet och som är lika eller större i högerträdet.

---

6)(1p) Som index i en hashtabell för tal kan man exempelvis ha tal % 10. Skissa på hur en sådan hashtabell ser ut då talen ovan i uppgift 5 instoppas i den i angiven ordning om kollisioner hanteras med öppen adressering och hoppfunktionen hopp = 1 och sedan hopp = hopp + 1.

---

7)(1p) Hur kommer hashtabellen, enligt uppgift 6 ovan att se ut, om man istället använder stackar för att hantera kollisioner?

---

8)(1p) Vad blir utskriften från följande program?

```
#include <stdio.h>
void rf(int n)
{
   if (n > 0)
   {
      printf("%d\n", n);
      rf(n-1);
      printf("%d\n", n);
   }
}

void main()
{
   rf(2);
}
```

---

9)(1p) Antag att du har en 8 bitars unsigned char definierad enligt:

```
unsigned char uch = 8;
```

Ange värdet för uch efter satsen:

```
uch |= (1 << 2);
```

---

10)(1p)Antag att du har följande vektor av strängar:

```
char *str[] = {"Hej", "på" ,"dig"};
```

Vad är `str[0][3]`?

11)(2p)En växelkassa innehållande mynt kan avbildas som en abstrakt datatyp enligt:

```
/* Specifikation -- Kassa.h */

typedef struct
{
   int femtio;    /* Antal femtioöringar */
   int en;        /* Antal enkronor */
   int fem;       /* Antal femkronor */
   int tio;       /* Antal tiokronor */
} kassa;

void las_kassa(kassa *kp);
/* Frågar efter och läser in kassa på formen femtio, en, fem, tio */

void skriv_kassa(kassa k);
/* Skriver ut kassa på skärmen på formen femtio, en, fem, tio */

float varde_kassa(kassa k);
/* Returnerar k:s värde i kr */

int mindre_kassa(kassa k1, kassa k2);
/* Sant om k1:s värde mindre än k2:s, annars falskt */

kassa summa_kassa(kassa k1, kassa k2);
/* Summakassan av k1 och k2 */
```

Implementera funktionen las_kassa.

---

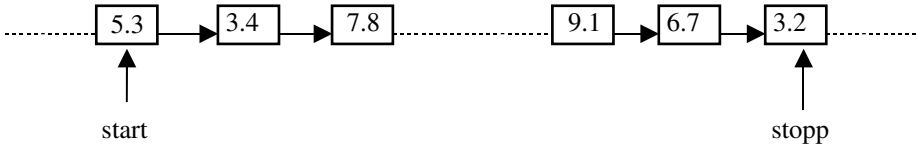
12)(2p)Implementera funktionen skriv_kassa, enligt uppgift 11 ovan.

13)(2p)Fullborda funktionen absolut, som ska returnera absolutbeloppet av tal. Inga jämförelseoperatorer finns tillgängliga för tal, så du måste kontrollera om den mest signifikanta biten är ettställd, då är talet negativt. Du kan anta att int är 32 bitar. Funktionshuvud enligt:

```
int absolut(int tal)
{
```

---

14)(2p)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 andra_start nedan, så att den flyttar fram start nr st länkar. Passeras stopp under denna förflyttning ska start stanna och peka på samma länk som stopp. Funktionen anropas exempelvis enligt:

```
andra_start(&start, stopp, 10);

void andra_start(linktyp **startpek, linktyp *stopp, int nr)
{
```

---

15)(2p)Skriv en rekursiv funktion som räknar antalet länkar i en envägslista av samma typ som i uppgift 3 ovan. Funktionshuvud enligt:

```
int nr_links(linktyp *lp)
{
```

16)(5p)Skriv ett fullständigt program som dynamiskt skapar tre heltalsvariabler, läser in värden till dessa och skriver ut det minsta värdet. Avslutningsvis ska programmet frigöra det dynamiskt allokerade minnet.

---

17)(5p)Skriv ett fullständigt program som upprepat slumpar ASCII-koder mellan 65 och 90 och lägger dessa på en stack. Upprepningen avslutas så fort ASCII-koden 90 (Z) slumpas. Programmet ska sedan tömma stacken samtidigt som de till ASCII-koderna hörande bokstäverna, A-Z, skrivs ut. Varje utskrift ska alltså börja med ett Z och sedan ska inga fler Z finnas med. För hantering av stacken ska du använda:

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

typedef char 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 */
```

---

18)(5p)Implementera funktionerna summa_kassa och varde_kassa, enligt uppgift 11 ovan och skriv ett huvudprogram som läser in två kassor, skriver ut summan av dessa och värdet av summakassan.

–

19)(5p)I textfilen kassor.txt finns ett antal kassor av den abstrakta datatypen kassa, enligt uppgift 11 ovan, på formen femtio, en, fem, tio radvis enligt:

```
5, 12, 6, 4
3, 14, 8, 2
2, 5, 2, 3
1, 1, 5, 8
1, 5, 2, 3
```

Skriv ett fullständigt program som läser alla kassor från filen och stoppar in dessa i en tvåvägslista sorterad enligt mindre_kassa-funktionen. Avslutningsvis ska programmet skriva ut de sorterade kassorna på skärmen med tillägg av värdet enligt:

```
1, 5, 2, 3 = 45.50
2, 5, 2, 3 = 46.00
3, 14, 8, 2 = 75.50
```

```
5, 12, 6, 4 = 84.50
1, 1, 5, 8 = 106.50
```

För hantering av tvåvägslistan ska du använda:

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

#include ”Kassa.h”
typedef kassa 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 omtentamen i Programmeringsmetodik, 5p, 020109

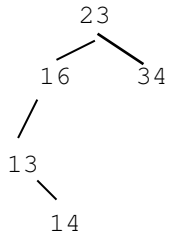
1)
```
*rp *= 2;
```

2)
```
free(lp);
lp = tp;
```

3)
```
tp->next = malloc(sizeof(linktyp));
tp = tp->next;
tp->data = 55;
tp->next = NULL;
```

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

5)



6)
```
3 -> 23
4 -> 34
5 -> 14
6 -> 16
7 NULL
8 NULL
9 -> 13
```

7)
```
3 -> 13 -> 23
4 -> 14 -> 34
5 NULL
6 -> 16
```

8)
```
2
1
1
2
```

9)
```
12
```

10) Tecknet '\0'

11)

```
#include "Kassa.h"
#include <stdio.h>

void las_kassa(kassa *kp)
{
   printf("Kassa på formen antal femtio, en, fem, tio ? ");
   scanf("%d, %d, %d, %d", &kp->femtio, &kp->en, &kp->fem, &kp->tio);
}
```

12)

```
void skriv_kassa(kassa k)
{
   printf("%d, %d, %d, %d", k.femtio, k.en, k.fem, k.tio);
}
```

13)

```
int absolut(int tal)
{
   if (tal & (1  << 31))
      return -tal;
   return tal;
}
```

14)

```
void andra_start(linktyp **startpek, linktyp *stopp, int nr)
{
   int i = 1;

   while (*startpek != stopp && i <= nr)
   {
      *startpek = (*startpek)->next;
      i++;
   }
}
```

15)

```
int nr_links(linktyp *lp)
{
   if (lp == NULL)
      return 0;
   else
      return 1 +  nr_links(lp->next);
}
```

16)

```
#include <stdio.h>
 #include <stdlib.h>
#include <conio.h>

void main()
{
   int *xp, *yp, *zp;

   xp = malloc(sizeof(int));
   yp = malloc(sizeof(int));
   zp = malloc(sizeof(int));


   printf("Tal1? ");
   scanf("%d", xp);

   printf("Tal2? ");
   scanf("%d", yp);

   printf("Tal3? ");
   scanf("%d", zp);


   if (*xp < *yp && *xp < *zp)
      printf("%d\n", *xp);
   else if (*yp < *zp)
      printf("%d\n", *yp);
   else
      printf("%d\n", *zp);


   free(xp);
   free(yp);
   free(zp);
   getch();
}
```

17)

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

void main()
{
   linktyp *lp = NULL;
   char ch;

   srand((unsigned)time(NULL));
   do
   {
      ch = rand() % 26 + 65;
      push(&lp, ch);
   } while (ch != 90);
```

```
   while (lp != NULL)
      printf("%c ", pop(&lp));
   getch();
}
```

18)

```
#include "Kassa.h"


kassa summa_kassa(kassa k1, kassa k2)
{
   kassa sum;

   sum.femtio = k1.femtio + k2.femtio;
   sum.en = k1.en + k2.en;
   sum.fem = k1.fem + k2.fem;
   sum.tio = k1.tio + k2.tio;
   return sum;
}

float varde_kassa(kassa k)
{
   return k.femtio*0.5 + k.en + k.fem*5 + k.tio*10;
}


/* Huvudprogram -- Kassmain.c */

#include <conio.h>
#include <stdio.h>


void main()
{
   kassa k1, k2, ksum;

   las_kassa(&k1);
   las_kassa(&k2);
   ksum = summa_kassa(k1, k2);
   skriv_kassa(ksum);

   printf("\n\nVärde summakassa : %.2f\n", varde_kassa(ksum));
   getch();
}
```

19)

```
/* Kasslist.c */

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

void main()
{
   FILE *tsin;
   headtyp *hp;
   linktyp *lp;
   kassa k;
   float varde;

   tsin = fopen("Kassor.txt", "r");
   newhead(&hp);
   while (fscanf(tsin, "%d, %d, %d, %d",
                 &k.femtio, &k.en, &k.fem, &k.tio)!= EOF)
   {
      newlink(&lp);
      putlink(k, lp);
      insort(lp, hp, mindre_kassa);
   }
   fclose(tsin);


   lp = firstlink(hp);
   while (lp != NULL)
   {
      k = getlink(lp);
      skriv_kassa(k);
      printf(" = %.2f\n", varde_kassa(k));
      lp = succlink(lp);
   }
   getch();
}
```