ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK
OCH NATURVETENSKAP

Namn:..... Klass:..... Pnr:.....

Omtentamen i Programmeringsmetodik C, 5p, D1 och E1, 000111

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 namn och personnummer på varje inlämnat extra papper.

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 .

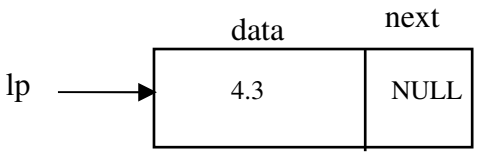
Detta häfte inlämnas.

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 byter ut det reella talet mot kvadratroten för talet.

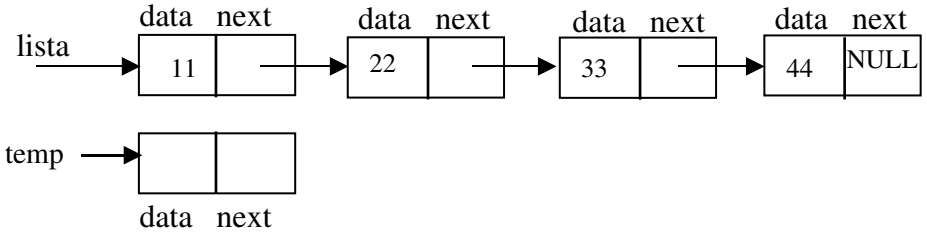
---

2)(1p) Skriv de satser som ger länkens delar värdena 4.3 och NULL enligt:



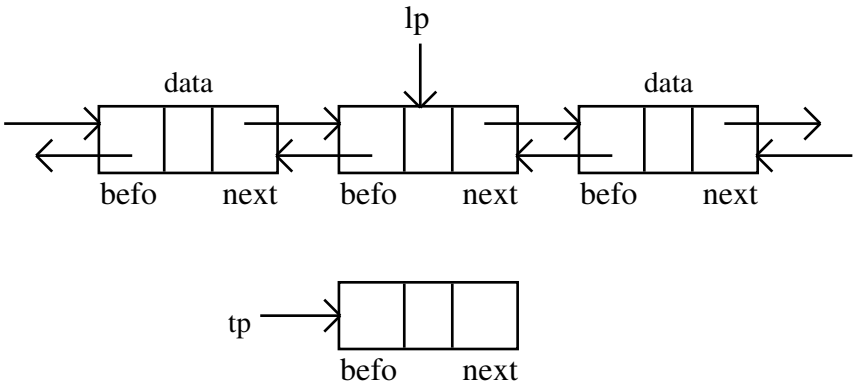
---

3) (1p) Skriv de satser som stoppar in länken som temp pekar på mellan de två första (vänstra) länkarna i en envägslista enligt figur. Listan ska hänga ihop efteråt och pekaren lista ska peka på samma länk som tidigare.



---

4) (1p) 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.



5) (1p) Rita det binära träd som skapas då heltalen 18, 30, 14, 20, 18, 12, 36 och 22 instoppas i trädet i den angivna ordningen, om instoppningsfunktionen sätter in data som är mindre i vänster träd och lika stora eller större i höger träd.

---

6)(1p) Ett binärt träd består av ett antal noder innehållande data, en vänster- och högerpekare till nya noder. Använd typedef för att definiera nodtyp för det binära trädet enligt uppgift 5 ovan.

---

7)(1p) Stoppa in heltalen enligt uppgift 5 ovan i angiven ordning i hashtabellen nedan om hashfunktionen tal % 5 används och kollisioner hanteras med LIFO-listor.

```
htab[0]
htab[1]
htab[2]
htab[3]
htab[4]
```

---

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

```
#include <stdio.h>
void rf(int n)
{
   printf("%d\n", n);
   if (n > 0)
      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 = 3;
```

Ange värdet för uch efter satsen:

```
uch <<= 2;
```

---

10)(1p)Sätt sp att peka på första tecknet i strängen str och skriv de satser som använder sp för att räkna antalet tecken i str.

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

11)(2p)En process som körs i en dator kan avbildas som en abstrakt datatyp enligt:

```
/* Specifikation av process - proc.h */
#include <stdio.h>

typedef
struct
{
   int pid;       /* Processens id */
   int prior;     /* Processens prioritet */
   float ptid;    /* Processens tid */
} process;

int skapa_process(process *pp, FILE *ptin);
/* Läser in data för en process från textströmmen ptin. Returnerar
som fscanf */

void visa_process(process p);
/* Skriver ut data för en process på skärmen */

float kor_process(process *pp, float ktid);
/* Minskar processtiden med ktid. Blir den resterande processtiden
negativ sätts den till 0.0. Returnerar den resterande processtiden */

int jfr_processer(process p1, process p2);
/* Returnerar sant om p1 har högre prioritet än p2. Vid samma
prioritet returneras sant om processtiden för p1 större än
processtiden för p2 */
```

Implementera funktionen kor_process.

---

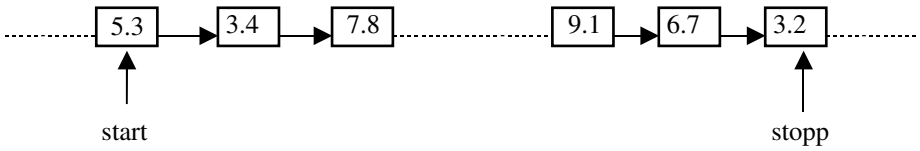
12)(2p)Implementera funktionen jfr_processer för den abstrakta datatypen process ovan.

13)(2p)Fullborda funktionen antal_noll som ska returnera antalet nollställda bitar i uch. Är exempelvis uch 'A' med ASCII-koden 01000001 ska funktionen returnera 6.

```
int antal_noll(unsigned char uch)
{
```

---

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 med_lista nedan, som tar start och stopp som parametrar och som returnerar medelvärdet av alla tal från och med start till och med stopp.

```
float med_lista(linktyp *start, linktyp *stopp)
{
```

---

15)(2p)Skriv en rekursiv funktion som räknar antalet löv i ett binärt träd. Ett löv är en nod som har värdet NULL på både left- och rightpekaren. Funktionshuvud enligt:

```
int leaves_tree(nodtyp *rp)
{
```

16)(5p)Implementera alla funktioner för nedanstående abstrakta datatyp som avbildar rationella tal enligt:

```
/* Specifikation av rationella tal -- rtal.h */
typedef
struct
{
   int t;   /* Täljare */
   int n;   /* Nämnare */
} rtal;

void las_rtal(rtal *rp);
/* Läser in ett rationellt tal från tangentbordet */

void skriv_rtal(rtal r);
/* Skriver ett rationellt tal på skärmen */

rtal mul_rtal(rtal r1, rtal r2);
/* Multiplicerar två rationella tal */

int jfr_rtal(rtal r1, rtal r2);
/* Sant om r1 < r2 */
```

---

17)(5p)Skriv ett program som dynamiskt allokerar en vektor med ett inläst antal element av rationella tal enligt ovan. Efter inläsning av elementens värden ska dessa skrivas ut i omvänd ordning. Körexempel, där du matar in det understrukna:

```
Ge antal rationella tal : 3
Ge bråk på formen t/n : 1/2
Ge bråk på formen t/n : 2/3
Ge bråk på formen t/n : 3/4
3/4 2/3 1/2
```

---

18)(5p)I binärfilen sermat.dat finns ett antal serier av reella tal. Varje serie avslutas med 0.0 följt av seriens heltalsnummer. Skriv ett program som läser talen i varje serie från filen, stoppar in talen på en stack och skriver ut seriens nummer följt av talen enligt:

```
13
1.19 1.27 1.43 1.32 1.27
1.56 1.32 1.43 1.25 1.65
1.34 1.23 1.56 1.32 1.23
1.12 1.45 1.34 1.23

35
2.35 2.43 2.34 2.41 2.26
2.52 2.21 2.62 2.56 2.38
2.45 2.34

26
3.52 3.25 3.45 3.43 3.26
3.19 3.72 3.46 3.27 3.56
3.37 3.51 3.21 3.54 3.28
3.45 3.32 3.64 3.45 3.18
3.23 3.56
```

För hantering av stacken ska du använda en LIFO-lista enligt specifikation på nästa sida.

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

typedef float 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 och returnerar data från LIFO-listan */
```

---

19)(5p)I textfilen process.txt finns ett antal processer, enligt uppgift 11 ovan, sparade med processid, prioritet och körtid radvis enligt:

```
1 5 6.5
2 4 7.0
3 4 6.0
.......
```

Skriv ett program som läser alla processer från filen och sätter in processerna i en processkö i form av en tvåvägslista sorterad enligt jämförelsefunktionen. Efter instoppning av alla processer i kön, plockas processerna bort från kön, skrivs ut på skärmen och körs i 2.5 tidsenheter. Har en process körtid kvar ska den stoppas tillbaka i kön sorterad enligt jämförelsefunktionen. Detta ska fortsätta tills processkön är tom. För hantering av tvåvägslistan ska du använda:

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

#include ”proc.h”
typedef process 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 C, 5p, D1 och E1, 000111

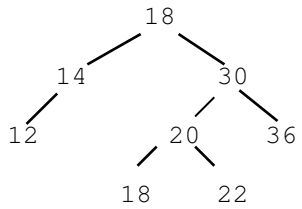
1)
```
*rp = sqrt(*rp);
```

2)
```
lp->data = 4.3;
lp->next = NULL;
```

3)
```
temp->next = lista->next;
lista->next = temp;
```

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

5)



6)
```
typedef
struct nod
{
   int data;
   struct nod *left, *right;
} nodtyp;
```

7)
```
htab[0] -> 20 -> 30
htab[1] -> 36
htab[2] -> 22 -> 12
htab[3] -> 18 -> 18
htab[4] -> 14
```

8)
```
2
1
0
0
1
2
```

9)
```
12
```

10)
```
int len = 0;
sp = str;
while (*sp != '\0')
{
   len++;
   sp++;
}
```

11)
```
#include ”proc.h”

float kor_process(process *pp, float ktid)
{
   pp->ptid -= ktid;
   if(pp->ptid < 0.0)
      pp->ptid = 0.0;
   return pp->ptid;
}
```

12)
```
#include ”proc.h”

int jfr_processer(process p1, process p2)
{
   if (p1.prior > p2.prior)
      return 1;
   else if(p1.prior == p2.prior && p1.ptid > p2.ptid)
      return 1;
   return 0;
}
```

13)
```
int antal_noll(unsigned char uch)
{
   int i, sum = 0;

   for (i = 0; i <= 7; i++)
   {
      if ((uch & (1 << i)) == 0)
         sum++;
   }
   return sum;
}
```

14)
```
float med_lista(linktyp *start, linktyp *stopp)
{
   linktyp *temp;
   float sum = 0.0;
   int nr = 0;

   temp = start;
   while (temp != stopp)
   {
      nr++;
      sum += temp->data;
      temp = temp->next;
   }
   nr++;
   sum += temp->data;
   return sum/nr;
}
```

15)
```
int leaves_tree(nodtyp *rp)
{
   if (rp == NULL)
      return 0;
   else if (rp->left == NULL && rp->right == NULL)
      return 1;
   return leaves_tree(rp->left) + leaves_tree(rp->right);
}
```

16)

```
/* Implementation av rationella tal -- rtal.c */

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

void las_rtal(rtal *rp)
{
   printf("Ge bråk på formen t/n : ");
   scanf("%d/%d", &rp->t, &rp->n);
}

void skriv_rtal(rtal r)
{
   printf("%d/%d ", r.t, r.n);
}

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

   r.t = r1.t * r2.t;
   r.n = r1.n * r2.n;
   return r;
}

int jfr_rtal(rtal r1, rtal r2)
{
   return (float)r1.t/r1.n < (float)r2.t/r2.n;
}
```

17)

```
/* Huvudprogram -- dynrtal.c */

#include "rtal.h"
#include <stdlib.h>
#include <stdio.h>

void main()
{
   rtal *rvek;
   int nr, i;

   printf("Ge antal rationella tal : ");
   scanf("%d", &nr);
   rvek = calloc(nr, sizeof(rtal));
   for ( i = 0; i < nr; i++)
   {
      las_rtal(&rvek[i]);
   }

   for ( i = nr - 1; i >= 0; i--)
   {
      skriv_rtal(rvek[i]);
   }
   free(rvek);
}
```

18)

```
/* Huvudprogram -- sermat.c */

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

void main()
{
   FILE *bin;
   linktyp *lp = NULL;
   float x;
   int sernr, nr;

   bin = fopen("sermat.dat","rb");
   fread(&x, sizeof(float), 1, bin);
   while (!feof(bin))
   {
      if (x == 0.0)
      {
         fread(&sernr, sizeof(int), 1, bin);
         nr = 0;
         printf("%d", sernr);
         while (lp != NULL)
         {
            if (nr % 5 == 0)
               putchar('\n');
            nr++;
            printf("%.2f ", pop(&lp));
         }
         printf("\n\n");
      }
      else
         push(&lp, x);
      fread(&x, sizeof(float), 1, bin);
   }
}
```

19)

```
/* Huvudprogram -- procmain.c */


#include "twolist.h"

void main()
{
   FILE *tin;
   process p;
   headtyp *prochead;
   linktyp *proclink;

   tin = fopen("Process.txt","r");
   newhead(&prochead);
   while (skapa_process(&p, tin) != EOF)
   {
      newlink(&proclink);
      putlink(p, proclink);
      insort(proclink, prochead, jfr_processer);
   }
   fclose(tin);

   proclink = firstlink(prochead);
   while (proclink != NULL)
   {
      outlist(proclink);
      p = getlink(proclink);
      visa_process(p);
      if (kor_process(&p, 2.5) > 0.0)
      {
         putlink(p, proclink);
         insort(proclink, prochead, jfr_processer);
      }
      proclink = firstlink(prochead);
   }
}
```