UiT

NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	Inf-1100
Dato:	Tirsdag 25. februar 2014
Tid:	Kl 09:00 - 13:00
Sted:	Aud.Max.

Tillatte hjelpemidler: Ingen

Eksamen inneholder 4 sider
inklusiv denne forside

Kontaktperson:	Anders Andersen
Telefon:	95180675

UNIVERSITETET I TROMSØ

Les godt gjennom hele oppgavesettet før du begynner å løse oppgavene. Der oppgaven ber deg lage eller implementere en funksjon så kan du bruke C-liknende pseudokode. Du kan også referere tilbake til funksjoner du tidligere har skissert. Funksjonen `sqrt()` kan benyttes for å beregne kvadratroten av et tall der det er nødvendig.

Oppgave 1 – 25%

De fleste av dagens datamaskiner er strukturert i henhold til en modell beskrevet av John von Neumann i 1946. Beskriv komponentene i denne modellen og kommunikasjonen mellom disse.

Oppgave 2 – 20%

Rekken av *Fibonaccitall* $(0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots)$ er definert slik:

$$\begin{aligned} F(1) &= 0 \\ F(2) &= 1 \\ F(n) &= F(n-2) + F(n-1) \end{aligned}$$

Det vil si at det første tallet i rekken er 0. Det andre er 1, og deretter er hvert tall definert som summen av de to foregående tallene

a) Lag en funksjon som returnerer det n'te Fibonaccitallet:

```
int fib(int n);
```

b) Lag en funksjon som returnerer antall Fibonaccitall mindre enn tallet t:

```
int numfib(int t);
```

c) Lag en funksjon som returnerer det største Fibonaccitallet som er mindre enn tallet t:

```
int prefib(int t);
```

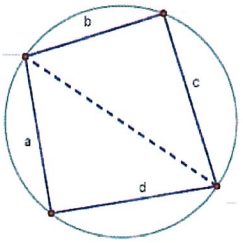
(Hint: Bruk funksjonene fra deloppgavene over.)

Oppgave 3 – 25%

Fra Brahmaguptas formel så har vi at arealet A til en quadrilateral (som firkanten i figuren) er følgende:

$$A = \sqrt{(s-a)(s-b)(s-c)(s-d)},$$

$$\text{hvor } s = \frac{a+b+c+d}{2}$$



Vi kan benytte Herons formel til å regne arealet av en trekant med sidene a, b og c:

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \quad \text{hvor } s = \frac{a+b+c}{2}$$

a) Lag en funksjon som beregner arealet til en quadrilateral gitt lengden på de fire sidene:

```
float brahmagupta(float a, float b, float c, float d);
```

b) Lag en funksjon som beregner arealet til en trekant gitt lengden på de tre sidene:

```
float heron(float a, float b, float c);
```

c) Gitt en datastruktur som spesifiserer koordinatene til hjørnene i en firkant:

```
typedef struct firkant firkant_t;
struct firkant {
  float x1, y1;
  float x2, y2;
  float x3, y3;
  float x4, y4;
};
```

Lag en funksjon som beregner arealet til en firkant. Funksjonen skal ta som inn-parametre en peker til en firkant datastruktur (`firkant_t*`) og returnere arealet til firkanten:

```
float areal(firkant_t *k);
```

Du kan ikke anta at firkanten er et kvadrat, men du kan anta at den er konveks (vinkelen mellom alle de indre kantene er mindre enn 180 grader). Bruk Pythagoras formel for å regne ut avstanden d mellom to punkter (x_1, y_1) og (x_2, y_2):

$$d = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

(Hint: Du kan bruke funksjonen som regner ut arealet til en trekant fra deloppgaven ovenfor i løsningen.)

Oppgave 4 – 30%

I denne oppgaven skal vi håndtere lister med et ukjent antall elementer. Hvert element har en nøkkel (verdi), og det eksisterer en funksjon `sammenlign()` som tar to elementer som argument og returnerer en verdi som indikerer forholdet mellom nøklene:

```
int sammenlign(void *e1, void *e2);
```

Funksjonen vil returnere -1 hvis nøkkel til `e1` har mindre verdi enn nøkkel til `e2`, 0 hvis nøkkel til `e1` har samme verdi som nøkkel til `e2`, og 1 hvis nøkkel til `e1` har større verdi enn nøkkel til `e2`.

a) Gitt to lister `a` og `b` som begge er sortert og med minste element først i listene. Lag en funksjon som tar `a` og `b` som argument og returnerer en ny liste med elementene fra `a` og `b` i sortert rekkefølge (fra minste til største nøkkel).

```
void merge(list_t *a, list_t *b);
```

Hvis to elementer har samme nøkkel så skal kun et av elementene i ny liste. Altså, ny liste skal ikke inneholde to eller flere elementer med samme nøkkel.

b) Gitt de samme to listene `a` og `b`. Lag en funksjon som tar `a` og `b` som argument og returnerer antall unike elementer tilsammen i disse to listene (altså antall unike nøkler):

```
int mergenum(list_t *a, list_t *b);
```

Du kan anta at følgende listefunksjoner er tilgjengelige:

```
// Lag en ny liste
list_t *list_create(void);

// Fjern og returner første element i en liste
void *list_removefirst(list_t *list);

// Sett inn et element sist i listen
int list_addlast(list_t *list, void *item);

// Lag en ny listeiterator som peker på første element i listen
list_iterator_t *list_createiterator(list_t *list);

// Frigi listeiterator
void list_destroyiterator(list_iterator_t *iter);

// Returner element som pekes på av iterator og la iterator peke på
// neste element (returner NULL når vi er kommet til enden av lista)
void *list_next(list_iterator_t *iter);
```