Eksamen INF-1100
Innføring i programmering
Høst 2010

Eksamenssettet består av 4 oppgaver.

Der oppgaven ber om at du skriver en funksjon kan du bruke C lignende pseudo-kode. Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

Oppgave 1 - 20%

a) Gi en kort beskrivelse av komponentene i von Neumann modellen.

b) Beskriv hvordan en instruksjon utføres i en datamaskin strukturert i henhold til von Neumann modellen. Fokuser på de ulike fasene i en instruksjonssyklus ('instruction cycle').

c) Beskriv kort hvordan I/O utføres i en von Neumann-basert datamaskin. Fokuser på hvordan et program kommuniserer med en I/O-enhet, samt på hvordan data overføres mellom minnet og en I/O-enhet.

Oppgave 2 - 20%

Gitt ett array med tilfeldige heltall. Skriv en funksjon *kMinste* som flytter rundt på tallene slik at de k minste tallene ligger først i arrayet. Du kan anta at k er mindre eller lik antall tall i arrayet. Funksjonen skal ta som inn-parametre en peker til arrayet, en angivelse av antall tall i arrayet, samt en verdi for k:

```
void kMinste(int *array, int arraystørrelse, int k);

Løsningsforslag 2:

void kMinste(int *array, int arraystørrelse, int k)
{
    int i, j;
    int min;
    int tmp;

    // Bruker selection sort, men hvor en stopper når k tall er sortert
    for (i = 0; i < k; i++) {
        min = i;
        for (j = i+1; j < arraystørrelse; j++) {
            if (array[j] < array[min])
                min = j;
        }
        // Bytter element i posisjon array[min] med element i posisjon array[i]
        tmp = array[i];
        array[i] = array[min];
        array[min] = tmp;
    }
}
```

Oppgave 3 - 20%

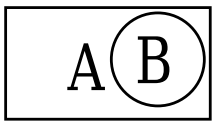
Skriv en funksjon *Forkort* som reduserer lengden på en tekststreng ved å erstatte repeterende tegn med $\{nX\}$, hvor n er antall tegn og X er selve tegnet. Funksjonen skal kun utføre erstatting ved mer enn 4 repeterende tegn. Funksjonen skal ta som inn-parametre en peker til en input tekststreng og en peker til en output tekststreng:

```
void Forkort(char *innstreng, char *utstreng);
```

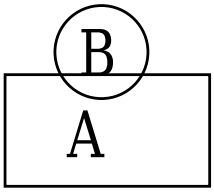
For eksempel, dersom innstreng er 'AAbbbbbbCCCCCCC', skal funksjonen skrive 'AA{6b}{7C}' til utstreng. I dette eksempelet er lengden på tekststrengen redusert fra 15 til 10 tegn. Husk at alle tekststrenger i C avluttes med verdien 0.

Løsningsforslag 3:

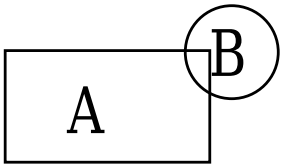
```
// Gjør en antagelse om at det eksisterer en funksjon sprintf som kan brukes
// for å skrive '{nX}' til utstreng.  Denne funksjonen ligner på printf,
// men skriver til en angitt tekststreng fremfor til terminalvinduet.
// sprintf returnerer antall tegn skrevet.

void Forkort(char *innstreng, char *utstreng)
{
    int i;
    int antall;
    int innpos, utpos;

    innpos = 0;
    utpos = 0;
    while (innstreng[innpos] != 0) {
        // Tell repetisjoner
        antall = 1;
        for (i = innpos + 1; innstreng[i] == innstreng[innpos]; i++)
            antall++;

        if (antall > 4) {
            utpos += sprintf(utstreng, ``{%d%c}'', antall, innstreng[innpos]);
            innpos += antall;
        } else {
           // Kopier fra innstreng til utstreng
           for (i = 0; i < antall; i++) {
               utstreng[utpos] = innstreng[innpos];
               utpos++;
               innpos++;
           }
        }
    }
}
```



1. Sirkel omsluttet av rektangel



2. Rektangelkant krysser sirkel



3. Hjørne av rektangel inne i sirkel

Oppgave 4 - 40%

Gitt en datastruktur som spesifiserer midtpunkt og radius til en sirkel:

```
typedef struct sirkel sirkel_t;
struct sirkel {
    float x, y;
    float radius;
};
```

og en datastruktur som spesifiserer nedre venstre hjørne og høyde og bredde til et rektangel:

```
typedef struct rektangel rektangel_t;
struct rektangel {
    float x, y;
    float høyde, bredde;
};
```

(a) Skriv en funksjon *Overlapp* som avgjør om et rektangel overlapper med en sirkel. Funksjonen skal ta som inn-parametre en peker til en rektangel datastruktur A og en peker til en sirkel datastruktur B. Funksjonen skal returnere verdien 1 dersom A og B overlapper, og verdien 0 hvis ikke:

```
int Overlapp(rektangel_t *A, sirkel_t *B);
```

Figuren over illustrerer de forskjellige situasjonene funksjonen må ta hensyn til:

1. Hele sirkelen er omsluttet av rektangelet.
2. En rektangelkant krysser sirkelen.
3. Et eller flere hjørner av rektangelet er inne sirkelen.

For å forenkle oppgaven kan du anta at *Overlapp* kan gjøre bruk av to eksisterende funksjoner, *Omsluttet* og *Krysser*, for å avgjøre om henholdvis situasjon 1 eller 2 er inntruffet:

```
int Omsluttet(rektangel_t *A, sirkel_t *B);
int Krysser(rektangel_t *A, sirkel_t *B);
```

Omsluttet og *Krysser* returnerer verdien 1 dersom situasjonen er inntruffet, og verdien 0 hvis ikke.

Din oppgave er i hovedsak å avgjøre om situasjon 3 har inntruffet. Hint: Sjekk avstanden mellom sirkelens midtpunkt og hjørnene i rektangelet. Dersom denne avstanden er mindre enn sirkelens radius er hjørnet inne i sirkelen. For å beregne avstanden mellom to punkter $(x1, y1)$ og $(x2, y2)$ kan du benytte Pythagoras' formel:

$$avstand = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Løsningsforslag 4a:

```
// Beregn avstand mellom to punkter
float dist(float x1, float y1, float x2, float y2) {
    return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
}

int Overlapp(rektangel_t *A, sirkel_t *B)
{

    if (Omsluttet(A, B) || Krysser(A, B))
        return 1;

    // Sjekk om hjørner til rektangel A er innenfor sirkel B
    if (dist(A->x,             A->y,            B->x, B->y) <= B->radius ||
        dist(A->x + A->bredde, A->y,            B->x, B->y) <= B->radius ||
        dist(A->x + A->Bredde, A->y + A->høyde, B->x, B->y) <= B->radius ||
        dist(A->x,             A->y + A->høyde, B->x, B->y) <= B->radius)
        return 1;
    return 0;
}
```

(b) Skriv en funksjon *OverlappListe* som avgjør om et rektangel overlapper med en eller flere sirkler i en liste. Funksjonen skal ta som inn-parametre en peker til et rektangel A, samt en peker til en liste av pekere til sirkel datastrukturer. Funksjonen skal returnere en liste med pekere til sirkel datastrukturer som overlapper med rektangelet A:

```
list_t *OverlappListe(rektangel_t *A, list_t *sirkelliste)
```

Du kan anta at funksjonen *Overlapp* fra oppgave (a) er tilgjengelig, samt at følgende listefunksjoner er tilgjengelige:

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

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

// Lag en ny listeiterator
list_iterator_t *list_createiterator(list_t *list);

// Returner element som pekes på av iterator og
// la iterator peke på neste element
void *list_next(list_iterator_t *iter);

// Frigi minne brukt av iterator
void list_destroyiterator(list_iterator_t *iter);
```

Løsningsforslag 4b:

```
list_t *OverlappListe(rektangel_t *A, list_t *sirkelliste)
{
    list_t *new;
    list_iterator_t *iter;
    sirkel_t *sirkel;

    new = list_create();
    iter = list_createiterator(sirkelliste);
    sirkel = list_next(iter);
    while (sirkel != NULL) {
        if (Overlapp(A, sirkel)) {
            list_addlast(new, sirkel);
        }
        sirkel = list_next(iter);
    }
    list_destroyiterator(iter);
    return new;
}
```