Eksamen INF-1100
Innføring i programmering
Høst 2007
Med løsningsforslag

Eksamenssettet består av 3 oppgaver.

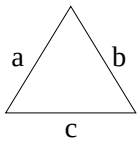
Der oppgaven ber om at du skriver en funksjon kan du bruke C lignende pseudo-kode. Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

Oppgave 1 - 30%

De fleste av dagens datamaskiner er strukturert i henhold til en modell foreslått av John Von Neumann i 1946. Beskriv denne modellen. Beskrivelsen bør omfatte de ulike komponentene i modellen og hvordan disse interagerer med hverandre.

Oppgave 2 - 40%

Gitt en trekant hvor a, b, og c angir lengdene på sidene i trekanten:



Herons formel kan benyttes for å beregne arealet til trekanten:

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

(a) Skriv en funksjon som beregner arealet av en trekant basert på Herons formel. Funksjonen skal ta som inn-parametre lengdene på sidene i trekanten og returnere arealet til trekanten. Du kan anta at lengdene er angitt som flyttall (*float*). Funksjonen *sqrt* kan benyttes for å beregne kvadratroten av et tall.

Løsningsforslag 2a:

```
float heron(float a, float b, float c) {
    float s = (a + b + c) / 2;
    return sqrt(s * (s - a) * (s - b) * (s - c));
}
```

(b) 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;
};
```

Skriv en funksjon som beregner arealet av en firkant. Funksjonen skal ta som inn-parameter en peker til en firkant datastruktur (*firkant_t**) og returnere arealet til firkanten. Du kan ikke anta at firkanten er et rektangel. Du kan anta at firkanten er konveks (vinkelen mellom alle de indre kantene er mindre enn 180 grader). 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 2b:

```
float FirkantAreal(firkant_t *q) {
float a1, a2, a3, b1, b2, b3;

// Finn lengde til kanter i første trekant
a1 = sqrt((q->x4 - q->x3) * (q->x4 - q->x3) + (q->y4 - q->y3) * (q->y4
- q->y3));
a2 = sqrt((q->x4 - q->x2) * (q->x4 - q->x2) + (q->y4 - q->y2) * (q->y4
- q->y2));
a3 = sqrt((q->x3 - q->x2) * (q->x3 - q->x2) + (q->y3 - q->y2) * (q->y3
- q->y2));

// Finn lengde til kanter i andre trekant
b1 = sqrt((q->x4 - q->x2) * (q->x4 - q->x2) + (q->y4 - q->y2) * (q->y4
- q->y2));
b2 = sqrt((q->x4 - q->x1) * (q->x4 - q->x1) + (q->y4 - q->y2) * (q->y4
- q->y2));
b3 = sqrt((q->x2 - q->x1) * (q->x2 - q->x1) + (q->y2 - q->y1) * (q->y2
- q->y1));

// Kalkuler areal vha herons formel
return heron(a1, a2, a3) + heron(b1, b2, b3);;
}
```

(c) Med utgangspunkt i firkant datastrukturen i oppgaven over, skriv en funksjon som fyller en firkant med en farge. Her kan du anta at følgende to funksjoner er tilgjengelige:

```
// Sett en piksel på skjermen
void SetPixel(int x, int y, unsigned int farge);

// Returner fargen til en piksel på skjermen
unsigned int GetPixel(int x, int y);

// Sett pixlene langs linjen fra x1,y1 til x2,y2
void DrawLine(int x1, int y1, int x2, int y2, unsigned int farge);
```

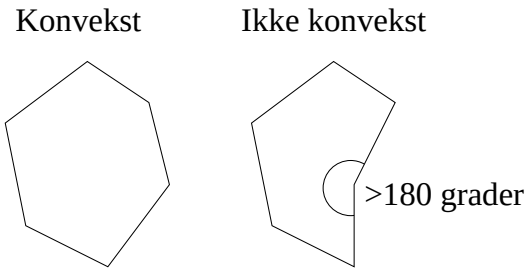
Løsningsforslag 2c:

```
void FyllFirkant(firkant_t *q) {
    int y, start, stop;
    int xmin, ymin, xmax, ymax;

    // Kalkuler bounding box
    xmin = min4(q->x1, q->x2, q->x3, q->x4);
    ymin = min4(q->y1, q->y2, q->y3, q->y4);
    xmax = max4(q->x1, q->x2, q->x3, q->x4);
    ymax = max4(q->y1, q->y2, q->y3, q->y4);

    // Tegner et omriss av firkant med PENCOLOR farge
    DrawLine(q->x1, q->y1, q->x2, q->y2, PENCOLOR);
    DrawLine(q->x2, q->y2, q->x3, q->y3, PENCOLOR);
    DrawLine(q->x3, q->y3, q->x4, q->y4, PENCOLOR);
    DrawLine(q->x4, q->y4, q->x1, q->y1, PENCOLOR);

    for (y = ymin; y <= ymax; y++) {
        for (start = xmin; GetPixel(start, y) != PENCOLOR; start++) ;
        for (stop = xmax; GetPixel(stop, y) != PENCOLOR; stop--) ;
        DrawLine(start, y, stop, y, FILLCOLOR);
    }
}
```

(d) Følgende datastrukturer spesifiserer koordinatene til hjørnene i et polygon:

```
typedef struct hjørne hjørne_t;
struct hjørne {
    float x;
    float y;
};

typedef struct polygon polygon_t;
struct polygon {
    int      antallhjørner; // Antall hjørner i polygonet
    hjørne_t *hjørne;       // Array av hjørnespesifikasjoner
};
```

Skriv en funksjon som beregner arealet av et polygon. Funksjonen skal ta som inn-parameter en peker til en polygon datastruktur (*polygon_t**) og returnere arealet til polygonet. Du kan anta at polygonet er konvekst (vinkelen mellom alle de indre kantene er mindre enn 180 grader):



Hint: Tenk inndeling i trekanter.

Løsningsforslag 2d:

```
float dist(Vertex *v1, Vertex *v2) {
    return sqrt((v2->x - v1->x) * (v2->x - v1->x) + (v2->y - v1->y) * (v2->y - v1->y));
}


float PolyArea(Polygon *p) {
    float area;
    float a, b, c;
    int i;
    Vertex v0, v1, v2;

    area = 0;
```

```
    // Første hjørne
    v0 = p->vertex[0];

    for (i = 1; i + 1 < p->nvertices; i++) {
        // Neste par med hjørner
        v1 = p->vertex[i];
        v2 = p->vertex[i + 1];

        // Finn lengde på sider
        a = dist(&v0, &v1);
        b = dist(&v0, &v2);
        c = dist(&v1, &v2);

        // Kalkuler areal til trekant vha. herons formel
        area += heron(a, b, c);
    }

    return area;
}
```

Oppgave 3 - 30%

De fleste teksbehandligsverktøy har en hjelpefunksjon for såkalt utfylling ('auto completion') av ord. Denne funksjonen baserer seg på at brukeren taster inn de første bokstavene i et ord, trykker på en tast (feks. TAB), og får presentert et utvalg av ord, fra en ordliste, som begynner med de inntastede bokstavene. For eksempel, anta at brukeren taster inn "eks" og at ordlisten inneholder følgende ord: "eksamen", "eksempel", "C", "programmere", "debugge". Brukeren vil da bli presentert med ordene 'eksamen' og 'eksempel' som mulige utfyllingskandidater.

Skriv en funksjon *utfylling* som tar som inn-parametre en tekststreng som representerer de første bokstavene i et ord, samt en peker til en liste av tekststrenger (en ordliste):

```
list_t *utfylling(char *ord, list_t *ordliste)
```

Funksjonen skal returnere en ny liste. Denne listen skal inneholde de ord i ordlisten som begynner med bokstavene angitt i inn-parametret 'ord'. Du trenger ikke ta hensyn til problemer rundt små og store bokstaver. Du kan anta at elementene i ordlisten er nullterminerte tekststrenger av type *char**, og at følgende listefunksjoner er tilgjengelige:

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

// Frigi liste. Elementer i listen blir ikke frigitt
void list_destroy(list_t *list);

// Sett inn et element først i en liste
int list_addfirst(list_t *list, void *item);

// Lag en ny listeiterator
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
void *list_next(list_iterator_t *iter);
```

Løsningsforslag 3:

```
list_t *complete(char *word, list_t *wordlist) {
    char *listword;
    int i, equal;

    // Lag liste
    list_t *words = list_create();

    // Lag listeiterator
    list_iterator_t *iter = list_createiterator(wordlist);

    // Finn ord som matcher
    while ((listword = list_next(iter)) != NULL) {
        equal = 1;
        for (i = 0; word[i] != '\0'; i++) {
            if (tolower(word[i]) != tolower(listword[i])) {
                equal = 0;
                break;
            }
       }

       // Legg ord til liste dersom lik
       if (equal == 1) {
           list_addfirst(words, listword);
        }
   }

   // Return list
   return words;
}
```