UiT
NORGES
ARKTISKE
UNIVERSITET

LØSNINGSFORSLAG

For eksamen i: **INF-1100 Innføring i programmering og datamaskiners virkemåte**

Dato: **Mandag 23. februar 2015**

Oppgave 1 – 25%

Gi en kort beskrivelse av von Neumann modellen. Beskrivelsen bør omfatte de ulike komponentene i modellen, hvordan disse interagerer med hverandre, og hva som skjer når instruksjonene i et program utføres.

Se løsningsforslag fra tidligere år.

Oppgave 2 – 20%

Gitt følgende program:

```
#include <stdio.h>

int main(int argc, char **argv)
{
     int x = 42;
     char *y = "42";
     int z[] = { 4, 2 };

     if (y[0] == y[1]) {
         printf("%d\n", x + z[0]);
     }
     else {
         printf("%d\n", x / z[1]);
     }
     return 0;
}
```

a) Forklar hva som er forskjellen på variablene x, y, og z, med spesielt fokus på hvilken type variablene har og hvordan verdiene deres er representert i datamaskinens minne.

x er en integer, og ligger lagret i minne som en 2-er komplement bitstring. Siden x er en lokal variabel, ligger den allokert i stack-framen til funksjonen.

y er en peker, dvs. en minneadresse. Vi ser av typen at y peker på en char, som er en 8-bits integer (en byte) som ofte representerer ascii-koden til et tegn. Vi ser av initialiseringen at y her peker på en nullterminert streng "42" som er allokert av kompilatoren en plass på heapen, som tre suksessive bytes. Med andre ord er y[0] == '4', y[1] == '2', og y[2] == '\0'. Selve minneadressen til strengen ligger allokert lokalt på stacken, på samme måte som for x.

z er et array som er allokert lokalt, i stack-framen til funksjonen. Hvert element i arrayet er en integer, og ligger lagret på suksessive adresser i minnet. Vi ser av initialiseringen at z[0] == 4 og z[1] == 2.

Selv om x og y begge fundamentalt sett er heltall, så varierer det hvilke operasjoner man gjøre med de to variablene pga. de deklarerte typene. Man kan f.eks. multiplisere og dividere med x, mens y kan brukes med array-syntaks slik at y[N] tilsvarer den byten som ligger på minneadresse y+N. Det er heller ikke gitt at sizeof(x) == sizeof(y). Dette kan variere ut i fra hvillken maskinarkitektur man kompilerer mot.

b) Hvilket tall vil skrives ut på skjermen når programmet kjøres? Forklar hvordan du kommer frem til svaret ditt.

Det tallet som skrives ut er 21.

Dette er fordi den første if-testen evaluerer uttrykket til usant, siden y[0] == '4' og y[1] == '2'. Dermed blir else-clausen utført, som skriver ut verdien av uttrykket `x / z[1]`***. Siden x == 42 og z[1] == 2, blir svaret 21.***

Oppgave 3 – 30%

a) Oversett følgende tall fra desimal til binær representasjon:
   - 29
   - 50

   Riktig svar er 29 = 11101 og 50 = 110010. Se løsningsforslag fra tidligere år.

b) Adder sammen følgende binære tall og oversett resultatet til desimal representasjon:
   - 00001010 + 00100101
   - 00101010 + 00000011

   Riktig svar er 00101111 = 47 og 00101101 = 45. Se løsningsforslag fra tidligere år.

c) Skriv et program som «dekomponerer» et tall og skriver det ut som en sum av 2-er potenser. Tallet som skal dekomponeres skal oppgis som et ekstra argument på kommandolinja. Bruk funksjonen `atoi` for å konvertere fra tekststrenger til heltall:

```
int atoi(char *s);
```

(Om du ikke vet hvordan man bruker argumenter fra kommandolinja kan du komme videre ved å anta at du allerede har tallet lagret i en variabel.)

Eksempel: dersom programmet invokeres fra kommandolinja slik:

```
./dekomponer 429
```

Skal en utskrift produseres som for eksempel ser slik ut:

```
429 = 256 + 128 + 32 + 8 + 4 + 1
```

```
int main(int argc, char **argv)
{
    int i, n, first = 1;

    if (argc < 2)
        return 0;
    n = atoi(argv[1]);
    printf("%d = ", n);
    for (i = 1; i*2 <= n;)
        i *= 2;
    while (n > 0 && i > 0) {
        if (n >= i) {
            if (!first)
                printf(" + ");
            printf("%d", i);
            first = 0;
            n -= i;
        }
        i /= 2;
    }
    printf("\n");
    return 0;
}
```

Oppgave 4 – 25%

I et terningspill er en sekvens av `n` terningkast representert som et array av heltall, hvor hvert element i arrayet har en verdi mellom 1 og 6 som tilsvarer antall øyne på terningen i det aktuelle kastet.

a) Implementer en funksjon

```
int yatzee(int *A, int n);
```

som skal sjekke om alle de `n` terningkastene i `A` er like. Hvis alle er like, skal funksjonen returnere 1, og ellers skal den returnere 0.

```
int yatzee(int *A, int n)
{
    int i;

    if (n < 2)
        // trivielt tilfelle: færre enn 2 tall
        return 1;
    for (i = 1; i < n; i++) {
        if (A[i-1] != A[i])
            // fant to ulike nabotall
            return 0;
    }

    // fant ingen ulike nabotall, så dermed er alle like
    return 1;
}
```

b) Implementer en funksjon

```
void frekvens(int *A, int n, int *B);
```

som teller antall forekomster av de ulike verdiene i `A` og fyller inn resultatet i arrayet `B` slik at `B[1]` er antall enere, `B[2]` er antall toere, `B[3]` er antall treere, osv. Anta at `B` i utgangspunktet inneholder verdien 0 for alle elementer.

```
void frekvens(int *A, int n, int *B)
{
    int i;
    for (i = 0; i < n; i++)
        B[A[i]]++;
}
```

c) Funksjonen `frekvens` er også brukt på noe kryptisk vis i denne funksjonen:

```
int ukjent(int *A, int n)
{
    int B[7] = { 0 };
    int *C = calloc(n+1, sizeof(int)); // Array av n+1 tall
    int x;

    frekvens(A, n, B);
    frekvens(B, 7, C);
    x = C[n];
    free(C);
    return x;
}
```

Hva er det funksjonen `ukjent` gjør (hvilket formål har den)? Forklar hvordan funksjonen virker.

Funksjonen er en alternativ implementasjon av `yatzee()`. Den virker ved å først beregne fordelingen av terningverdier, og så beregne fordelingen av fordelingen. Dersom alle terningene er like vil det være 1 verdi som forekommer `n` ganger. Ellers 0 slike verdier.