UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	**INF-1100 Innføring i programmering og datamaskiners virkemåte**
Dato:	**25.02.2020**
Klokkeslett:	**09:00 – 13:00**
Sted:	**ADM-K1.04**
Tillatte hjelpemidler:	**Ingen**
Type innføringsark (rute/linje):	**WiseFlow**
Antall sider inkl. forside:	**5**
Kontaktperson under eksamen: Telefon/mobil:	**Robert Pettersen** **47 24 75 52**

Vil det bli gått oppklaringsrunde i eksamenslokalet?

JA, ca. kl.: 10:30

NB! Det er ikke tillatt å levere inn kladdepapir som del av eksamensbesvarelsen.

Hvis det likevel leveres inn, vil kladdepapiret bli holdt tilbake og ikke bli sendt til sensur.

UNIVERSITETET I TROMSØ

Eksamen INF-1100
Innføring i programmering og datamaskiners virkemåte
Høst 2019

Eksamenssettet består av 4 oppgaver.

Les oppgaveteksten grundig og disponer tiden slik at du får tid til å svare på alle oppgavene. I noen oppgaver kan det være nødvendig å tolke oppgaveteksten ved å gjøre noen antagelser - gjør i så fall rede for hvilke antagelser du har gjort, men pass på å ikke gjøre antagelser som trivialiserer oppgaven.

Der du skal utvikle eller beskrive en algoritme anbefales det at du først beskriver algoritmen på et høyt abstraksjonsnivå, f.eks. med figurer, før du går videre med detaljer og eventuell pseudokode. Dersom det spørres etter en implementasjon i C kreves det ikke 100% syntaktisk korrekt kode. Dersom det spørres etter pseudokode kan du kan også skrive ren C-kode om du ønsker.

Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

Oppgave 1 - 25%

Gi en kort beskrivelse av von Neumann modellen og instruksjonssyklusen. Beskrivelsen bør omfatte de ulike komponentene i modellen og hvordan disse interagerer med hverandre.

Oppgave 2 - 25%

Gitt følgende funksjon:

```
int main()
{
    int x = 128;
    char *y = "128";
    int z[] = { 1, 2, 8 };

    if (y[0] != z[0]) {
        printf("%d\n", x + z[2]);
    }
    else {
        printf("%d\n", x / z[2]);
    }
    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.

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

Oppgave 3 - 25%

Gitt et array A som inneholder n heltall.

a) Skriv en funksjon *organiser* som flytter rundt på tallene i A slik at oddetall kommer før partall.

```
void organiser(int *A, int n);
```

b) Skriv en funksjon *sorter* som organiserer tallene i A slik at tallene ligger i stigende rekkefølge.

```
void sorter(int *A, int n);
```

c) A har et *majoritetselement* dersom et tall forekommer mer enn $n/2$ ganger i A.

Følgende algoritme kan benyttes for å avgjøre om A har et majoritetselement. Anta at vi først sorterer A. Dersom A har et majoritetselement må dette tallet være i $A[n/2]$. Dersom antall forekomster av $A[n/2]$ i A er høyere enn $n/2$, er $A[n/2]$ et majoritetselement.

Skriv en funksjon *majoritetselement* som avgjør om A har et majoritetselement. Funksjonen skal returnere 1 dersom A har ett majoritetselement, og 0 dersom det ikke finnes. Her kan du gjenbruke sorteringsfunksjonen fra **b)**.

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

Oppgave 4 - 25%

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, 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**.

Du kan **ikke** gjøre antagelser om hvordan listene er implementert. Følgende listefunksjoner er tilgjengelige:

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

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

// Fjerner et element fra listen
void list_remove(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. NULL returneres
// når det ikke finnes noe neste element.
void *list_next(list_iterator_t *iter);

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