NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i IDATG2102 – Algoritmiske metoder

Faglig kontakt under eksamen:	**Frode Haug**
Eksamensdato:	**13.desember 2024**
Eksamenstid (fra-til):	**09:00-13:00 (4 timer)**
Hjelpemiddelkode/Tillatte hjelpemidler:	**I - Alle trykte og skrevne** (kalkulator er *ikke* tillatt)
Annen informasjon:	
Målform/språk:	**Bokmål**
Antall sider (inkl. forside):	**4**

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig X **2-sidig** ☐

sort/hvit X **farger** ☐

Skal ha flervalgskjema ☐

Kontrollert av:

_____

Dato Sign

Oppgave 1 (teori, 25%)

Denne oppgaven inneholder tre totalt uavhengige oppgaver fra pensum.

a) 8 Et av eksemplene i pensum leser et postfix-uttrykk og regner ut svaret.
Vi har postfix-uttrykket: 7 3 + 3 2 + * 5 6 * 8 5 * + +
Hva blir svaret ? Skriv/tegn opp stakkens innhold etter hver gang den er endret.

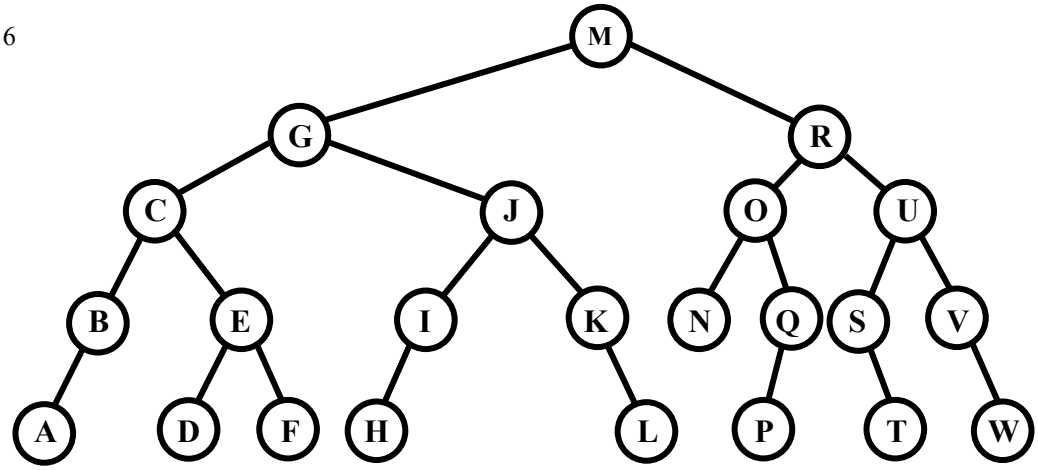
b) 8 Quicksort skal utføres på bokstavene/keyene: SJOKOMELK
Lag en oversikt/tabell der du for hver rekursive sortering skriver de involverte bokstavene og markerer/uthever hva som er partisjonselementet.

c) 9 Heapsort (vha. bottom-up heap konstruksjon) skal utføres på bokstavene/keyene
"S J O K O M E L K" (blanke regnes *ikke* med).
Skriv/tegn opp heapens innhold etterhvert som heapen konstrueres og deretter sorteres.

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre totalt uavhengige oppgaver fra pensum.

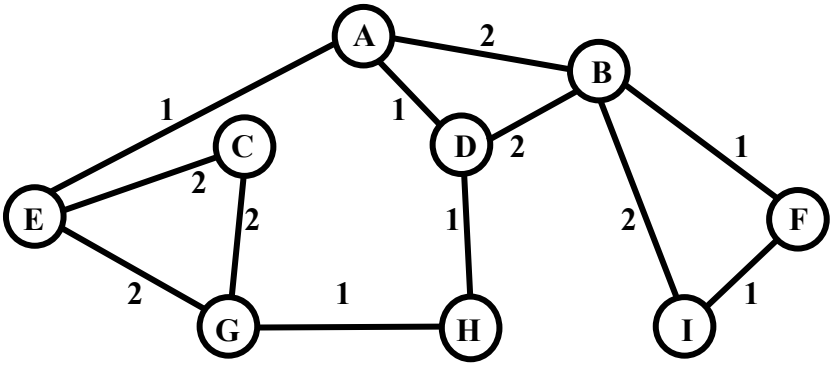
a) 6



Det skal fjernes («remove») noen noder fra det ovenfor gitte *binære søketreet*.
Skriv/tegn treet for hver gang, og fortell hvilken av «if ..... else if ..... else»-grenene i `EKS_28_BinertSokeTre.cpp` (dvs. Case 1, Case 2, Case 3) som er aktuelle når det etter tur fjernes henholdsvis bokstavene 'J', 'M' og 'R'.

NB: For hver fjerning skal det *på nytt* tas utgangspunkt i *hele* det aktuelle treet.
Dvs. *på intet tidspunkt* skal det fra treet være fjernet *mer enn en* bokstav.

b) 8 Følgende vektede (ikke-rettede) graf er gitt:



Vi bruker nabomatrise. Aktuell kode i et at pensumets eksempler utføres/kjøres på denne grafen.
Hvilke kanter er involvert i korteste-sti spenntreet fra noden D til *alle* de andre nodene? Skriv også opp innholdet i/på fringen etterhvert som koden utføres.
NB: *Husk at ved lik vekt så vil noden sist oppdatert (nyinnlagt eller endret) havne først på fringen ift. andre med den samme vekten.*

c) 11 Vis **konstruksjonsprosessen** når koden **for Huffman-koding** i EKS_39_Huffman.cpp brukes på teksten: OPTISKE ILLUSJONER ILLUSTRERER OPTISKE SAKER (inkludert de blanke). **Hvor mange bits trengs for å kode denne teksten?** Dvs. skriv/tegn opp:

- `frekvens`-arrayen
- `forelder`-arrayen
- Huffmans kodingstreet/-trien
- bokstavenes bitmønster (`kode`) og `lengde`
- totalt antall bits som brukes for å kode teksten

Oppgave 3 (koding, 40%)

Vi har et *binært søketre* bestående av nodene:

```
struct Node {
  int  ID;              //  Nodens ID/key/nøkkel/navn (et tall).
  Node *left,           //  Referanse til venstre subtre, eller nullptr/null (om tomt).
       *right;          //  Referanse til høyre subtre, eller:  se forklaring nedenfor.
  bool  harHoyreBarn;   //  Se forklaring nedenfor.
   Node (int id, Node* l, Node* r, bool hHB)
        { ID = id;  left = l;  right = r;  harHoyreBarn = hHB;  }
};
```

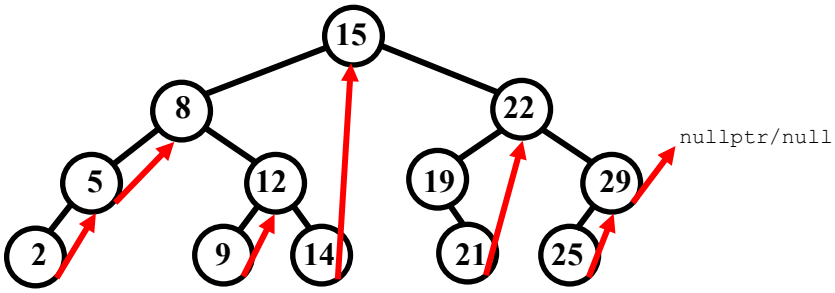
Vi har også den globale variabelen:

```
Node* gRoot = nullptr;      //  Rot-peker (har altså ikke at head->right er rota).
```

NB: Det skal i hele oppgave 3 *ikke* innføres noen flere globale variable, parametre, struct-medlemmer eller hjelpe-datastrukturer. Oppgavene kan løses vha. helt vanlig bruk av løkker og `if`-setninger.

I denne oppgaven er vi alltid interessert i *neste node i inorder rekkefølge*. Så om vi får en peker til en gitt node (kanskje midt inne i treet), skal det uansett være lett å finne neste node i inorder rekkefølge. Dette kan vi ivareta om vi bruker `right` på en litt spesiell måte: Om høyre subtre er tomt, så peker `right` på neste i *inorder rekkefølge*. Dette vil *alltid* være oppover et sted i treet!

Datamedlemmet `harHoyreBarn` er `false` dersom `right` er brukt på denne måten, og vil være `true` dersom `right` peker til et reelt høyre subtre. Dersom en node har tomt høyre subtre, og er den *aller* siste i treet, vil `harHoyreBarn` være `false` og `right` peke til `nullptr/null`. Eksempel på et slikt tre (der røde piler er når `right` er brukt på den spesielle måten):



a) 7 **Lag den ikke-relursive funksjonen** `Node* forste(Node* node)`
som returnerer en peker til den *sekvensielt første noden* (den med *minst* `id`) i subtreet under `node`. Er treet tomt skal den returnere `nullptr/null`.
NB: Dette er *ikke* det samme som den neste i inorder rekkefølge.

b) 9 **Lag den ikke-rekursive funksjonen** `bool finn(int id)`
som returnerer `true/false` til om `id` finnes i treet tilpekt av `gRoot`.

c) 8 **Lag den ikke-rekursive funksjonen** `void traverser(Node* node)`
som starter i `node` (som gjerne er midt inni treet et sted) og skriver ut *alle* `id`'ene i *hele* resten av treet. Husk at `node` også kan peke til `nullptr/null`.
NB: Husk på evt. å bruke tidligere lagd(e) funksjon(er).

d) 16 **Lag den ikke-rekursive funksjonen** `void legginn(int id)`
som legger inn en ny node (med ID lik 'id') etter prinsippene beskrevet ovenfor. Husk å håndtere tilfellet der treet er helt tomt, og at vi har constructoren, som gjør at man kan opprette nye noder vha. `new` og passende parametre. **Hint:** En ny node legges *alltid* inn nederst, pluss noen andre prinsipper som *alltid* vil gjelde (men de må du finne ut av selv).

Oppgave 4 (koding, 10%)

Vi har en lang tekst som inneholder `N` tegn (alle slags tegn på tastaturet, ikke bare bokstaver og sifre). Disse ligger i: `char txt[N]`. Det er *maksimum tre stykk* av samme tegn i arrayen. **Skriv et komplett program som skriver ut tegnene det er eksakt tre like av og hvor mange det dreier seg om.**
F.eks. for teksten: `hgfhnk!mBSgmiGtytAr4n8765!TASn68m5BrsseBgrT!GybbGtb798uyu76`
ville den ha kunnet skrevet ut noe slikt som:

```
! 6 7 8 B G b g m n r t y
Antall tegn det er tre like av: 13
```

Er det aktuelt å bruke noen kodesnutter fra pensum, så er det bare å henvise til dette.
NB: Det er *ikke* meningen at du her trenger å innføre noen hjelpe-array eller andre datastrukturer.

NB: I *hele* dette oppgavesettet skal du *ikke* bruke kode fra (standard-)biblioteker (slik som bl.a. STL og Java-biblioteket). Men de vanligste includer/import du brukte i 1.klasse er tilgjengelig. Koden kan skrives valgfritt i C++ eller Java.

Løkke tæll!
FrodeH