Høgskolen i Innlandet

KANDIDAT

118

PRØVE

2ADS101 1 Algoritmer og datastrukturer for spill

Emnekode	2ADS101
Vurderingsform	Skriftlig eksamen
Starttid	06.12.2021 08:00
Sluttid	06.12.2021 12:00
Sensurfrist	27.12.2021 22:59
PDF opprettet	15.02.2022 09:50

Seksjon 1

Oppgave	Tittel	Oppgavetype
ℹ	Forside	Informasjon eller ressurser

Oppgave 1 (30%)

Oppgave	Tittel	Oppgavetype
1	Oppgave 1a)	Langsvar
2	Oppgave 1b)	Langsvar
3	Oppgave 1c)	Langsvar
4	Oppgave 1d)	Langsvar
5	Oppgave 1e)	Langsvar

Oppgave 2 (30%)

Oppgave	Tittel	Oppgavetype
6	Oppgave 2a)	Programmering
7	Oppgave 2b)	Programmering
8	Oppgave 2c)	Programmering
9	Oppgave 2d)	Programmering

Oppgave 3 (15%)

Oppgave	Tittel	Oppgavetype
10	Oppgave 3a)	Langsvar
11	Oppgave 3b)	Langsvar

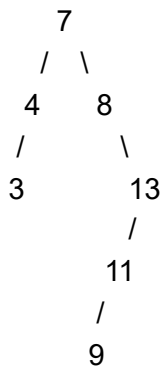
Oppgave 4 (25%)

Oppgave	Tittel	Oppgavetype
12	Oppgave 4a)	Programmering
13	Oppgave 4b)	Programmering
14	Oppgave 4c)	Programmering

1 Oppgave 1a)

(Oppgave 1a-e teller 30%) Gitt tallene 7, 8, 13, 4, 3, 11, 9. Tegn og forklar hvordan du bygger et binært søketre med tallene, satt inn i rekkefølge som ovenfor.

Skriv ditt svar her



Hvert tall blir satt inn i treet som en node. Hver node kan ha et venstre og et høyre barn.
7 blir satt inn som roten til treet. Da kommer 8 som blir høyre barnet til 7 fordi 8 er større enn 7.
13 går ned til høyre for 7 siden det er større enn både 7 og 8, 13 blir da høyre til 8. 4 blir venstre barnet til 7 siden det er mindre enn roten. 3 følger ned venstre og blir venstre barnet til 4.
11 er større enn 7 og 8,men mindre enn 13, så det blir da venstre barnet til 13. 9 følger 11 og blir venstre barnet dets.

Kandidat nr 118. Side 1

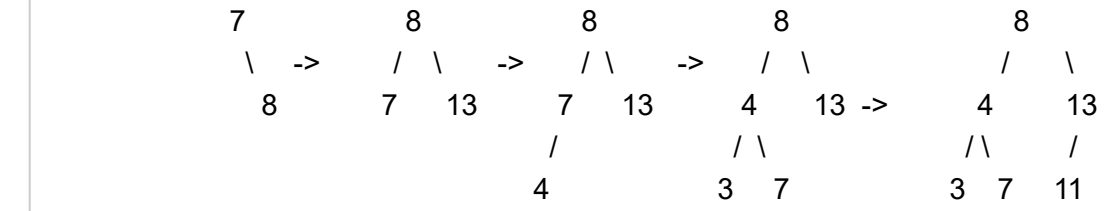
2 Oppgave 1b)

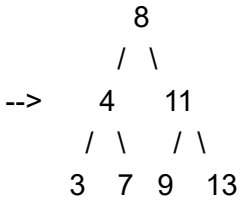
Tegn og forklar hvordan du bygger et AVL-tre med tallene, satt inn i rekkefølge som i forrige oppgave.

Skriv ditt svar her

Et AVL tree balanserer seg selv underveis og treet endres ofte ved innsetting av nye noder.

7, 8, 13, 4, 3, 11, 9





7 blir roten, 8 kommer og blir høyrebarnet siden 8 er større. Når 13 blir satt inn, blir 8 ny rot, for å holde treet balansert, 7 blir venstrebarnet og 13 blir høyrebarnet.

4 går ned venstre siden for roten og blir venstrebarn til 7. Når 3 blir satt inn blir 4 nytt venstrebarn til 8,3 blir venstre til 4 og 7 høyre.

For å holde treet balansert blir 11 satt inn som venstre til 13.

Når 9 blir satt inn blir 11 satt inn som høyre for 8, så 9 blir venstre barn til 11 og 13 høyrebarn.

Treet er da balansert når alle tallene er satt inn.

Kandidat nr 118. Side 2

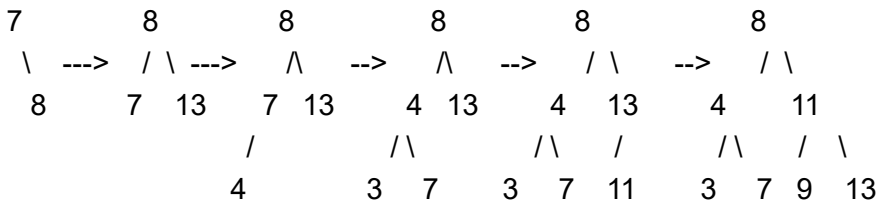
3 Oppgave 1c)

Tegn og forklar hvordan du bygger et Red-black tre med de samme tallene, satt inn i samme rekkefølge.

Skriv ditt svar her

7, 8, 13, 4, 3, 11, 9

Red-black treet blir innsatt på samme måte som AVL treet.



7 blir satt inn som rot, og er sort. 8 blir satt inn som høyre barn og som rød.

13 settes inn som rød, for å ha treet balansert, roterer vi, så nå er 8 rot, 7 venstre og 13 høyre, 7 og 8 bytter farge.

4 blir satt inn som sort, fordi 7 er rød. Den blir satt som venstre barn av 7

3 blir satt inn som rød som 4 sitt venstre barn, det er nå ubalansert. Gjør høyre rotasjon og bytter farge på 4 og 7.

11 blir satt inn som rød som venstre barn til 13.

9 blir satt innn som høyre barn av 11, som rød. For å balansere blir 11 nå høyrebarn av 8 og 13 høyrebarn av 11. 11 og 13 bytter farger. 7 blir sort.

Kandidat nr 118. Side 3

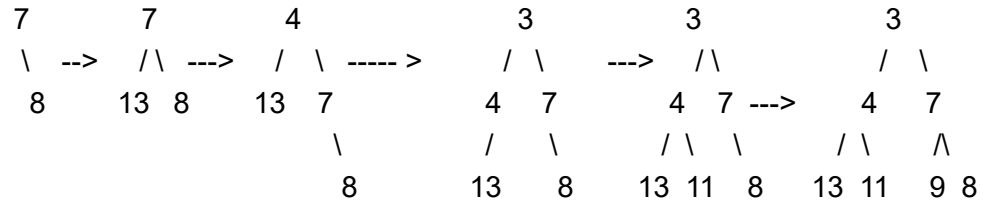
4 Oppgave 1d)

Tegn og forklar hvordan du bygger en min-heap med de samme tallene, satt inn i samme rekkefølge.

Skriv ditt svar her

I en min-heap skal de minste tallene være øverst i treet.

7, 8, 13, 4, 3, 11, 9



7 blir satt inn som roten. 8 blir da satt inn, siden 8 er større enn 7 kan det bli satt som enten høyre eller venstrebarn, høyre her.

13 er også større enn 7, så det blir satt som venstrebarn til 7.

4 blir satt inn som ny rot, 7 blir satt som høyre barn til 4, 8 følger med 7.

3 blir satt inn som ny rot siden det er mindre enn 4. 4 blir venstre barnet og 7 blir venstrebarnet.

11 går ned treet til høyre barn av 4, fordi det er større enn både 3 og 4.

9 blir venstre barn av 7 fordi det er større enn både 3 og 7, og for å holde treet balansert blir det satt til venstre barn av 7.

Kandidat nr 118. Side 4

5 Oppgave 1e)

Gitt en hashtabell med lengde 7, og kun en plass i hver skuff (bucket). Videre, gitt hashfunksjonene int hash(int key) { return key%7; }

og int dobbelthash(int key) { return 1+key%6; }

Tegn og forklar hvordan tallene blir plassert i hashtabellen. Forklar ut fra dette eksemplet hva slags klasing vi får her.

Skriv ditt svar her

0	1	2	3	4	5	6
7	8	13	4	3	11	9

Øverst er element plass, nede er verdien.

7 % 7 = 0, så 7 blir satt i element 0.

8 % 7 = 1, så element 1.

13%7 = 2, element 2.

4%7 = 3, element 3

3%7 = 4 , element 4

11 % 7 = 5, element 5

9 % 7 = 6, element 6.

Kandidat nr 118. Side 5

6 Oppgave 2a)

(Oppgave 2a-d teller 30%) I denne oppgaven skal du benytte nøyaktig de samme navnene som nedenfor. Følgende C++ kode er gitt:

```
class BinaryNode
{
private:
  char m_data;
  int m_frekvens;
  BinaryNode* m_left;
  BinaryNode* m_right;
public:
  explicit BinaryNode(char data) : m_data{data}, m_left{nullptr},
  m_right{nullptr}, m_frekvens{1} { }
  void insert(char data);
  void print() const;
};
```

Implementer funksjonen insert() slik at den setter inn en ny node som i et binært søketre, med følgende tilleggsbetingelse: Det er ikke tillatt med flere noder som har samme dataverdi. Dersom parameteren til insert er et tegn som allerede fins i en node, skal denne nodens m_frekvens inkrementeres.

Skriv ditt svar her

```
void BinaryNode::insert(char data)
{
//Kandidat nr 118. Side 6
    //Bruker std::queue til å ha alle nodene, for å så sjekke de
    std::queue<BinaryNode*> list;
    //Finn hvis det er noen lingende i treet
    list.push(this);
    BinaryNode* node;
    //Wont run if there is no right
    while (!list.empty()) {
        node = list.front();
        //Når pekeren til toppen er funnet, popper den fra queuen fordi vi er ferdig
        list.pop();
        //Hvis noden sin data er lik som data parameteren, legg til frekvensen
        if (node->m_data == data) {
            node->m_frekvens++;
        }
        //Hvis denne noden har barn, legg de til i queuen,
        if (node->m_left) {
            list.push(node->m_left);
        }
        if (node->m_right) {
            list.push(node->m_right);
        }
    }

    if (data < m_data) {
        if (m_left) {
            m_left->Insert(data);
        }
        else {
            left = new BinaryNode(data);
        }
    }
    else if (data > m_data) {
        if (m_right) {
            m_right->Insert(data);
        }
        else {
            m_right = new BinaryNode(data);

        }
    }


}
```

7 Oppgave 2b)

Implementer funksjonen print() slik at den skriver ut m_data og m_frekvens for alle nodene i treet når den kalles av roten.

Skriv ditt svar her

```
void BinaryNode::print() const
{
//Kandidat nr 118. Side 7
    if (m_left) {
        m_left->print();
    }
    std::cout << m_data << "=" << m_frekvens << ", ";
    if (m_right) {
        m_right->print();
    }
}
```

8 Oppgave 2c)

Skriv et lite testprogram hvor du anvender denne klassen og de to funksjonene på std::string s{"det er enklere å løse et algoritmisk problem enn et demokratisk problem"}.

Skriv ditt svar her

```
void main(){
    //Kandidat nr 118. Side 8
    std::string s{ "det er enklere å løse et algoritmisk problem enn et demokratisk

    //Danner roten som første element i stringen
    BinaryNode* node = new BinaryNode(s[0]);

    //Går gjennom hvert element i stringen og inserter elementet til binary treet
    for (int i = 1; i < s.size(); i++) {
        node->Insert(s[i]);
    }
    //Kaller print funksjonen
    node->Print();

    return 0;
}
```

9 Oppgave 2d)

Nodenes datadel består her av par av char og int (m_data og m_frekvens). Bruk samme testdata som i forrige oppgave og skriv kode som gir en sortert utskrift med hensyn på frekvens.

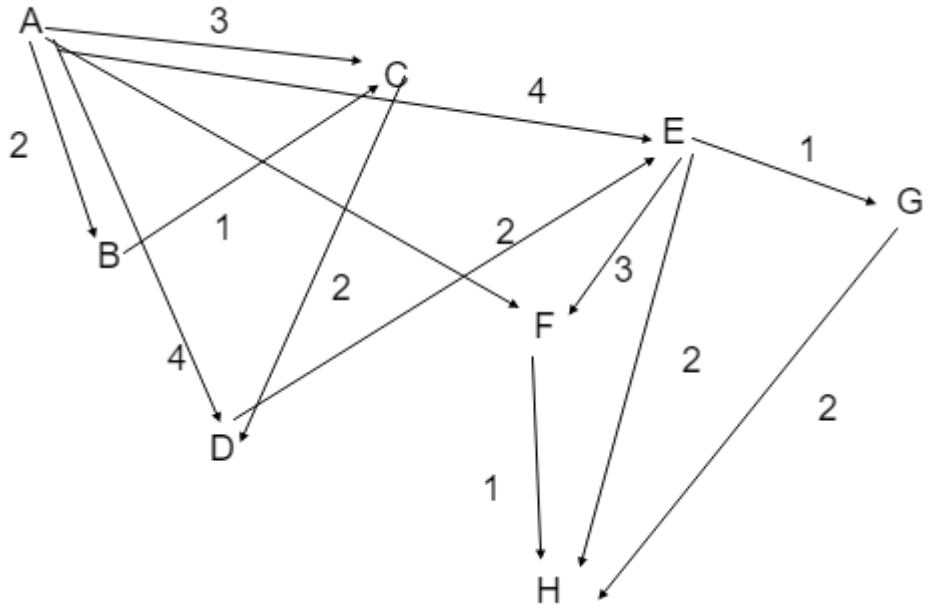
Skriv ditt svar her

```
//Overrider operatoren > til binary node, så når de sammenlignes vil de gå på hvem r
bool operator > (const BinaryNode& b1, const BinaryNode& b2){ return b1.m_frekvens >

void BinaryNode::PrintSorted()
{
//    //Kandidat nr 118. Side 9
    //Erklerer en kø for nodene, som sorteres ved den overkjørte operatoren, som gir
    std::priority_queue<BinaryNode*, std::vector<BinaryNode*>, std::greater<BinaryNo
    //En queue for å itterere gjennom alle nodene i treet og legge de til apq
    std::queue<BinaryNode*> list;
    //Pusher først seg selv inn inn i list
    list.push(this);
    //Erklerer en peker til en BinaryNode
    BinaryNode* node;
    while (!list.empty()) {
        node = list.front();
        list.pop();
        //Add it to the priorty queue
        apq.push(node);
        //Hvis denne noden har høyre eller vesntre barn, legg de til
        if (node->m_left) {
            list.push(node->m_left);
        }
        if (node->m_right) {
            list.push(node->m_right);
        }
        //Gjentas til det ikke er noen flere noder å finne
    }
    //Så lenge apq ikke er tom,
    while (!apq.empty()) {
        //Sett node til toppen, den laveste frekvensen
        node = apq.top();
        //Pop den ut av køen
        apq.pop();
        //Printer dataen og frekvensen
        std::cout << node->m_data << "=" << node->m_frekvens << ",  ";
    }
    std::cout << std::endl;

}
```

10 Oppgave 3a)

(Oppgave 3a-b teller 15%) En urettet graf består av nodene { A, B, C, D, E, F, G, H } og kantene { AB(2), AC(3), AD(2), AE(4), AF(4), BC(1), CD(2), DE(2), EF(3), EG(1), EH(2), FH(1), GH(2) }. Tegn og forklar hvordan du finner et minimum spenntre ved å bruke Kruskal's algoritme.

Skriv ditt svar her



Finner først de korteste veiene:
FH = 1, BC = 1, EG = 1.
Så de nest korteste
AB = 2, CD = 2, EH = 2, GH = 2, DE = 2. Jeg stryker da AC, AE, FH, GH, AD
Så videre
EF = 3, Stryker AF,

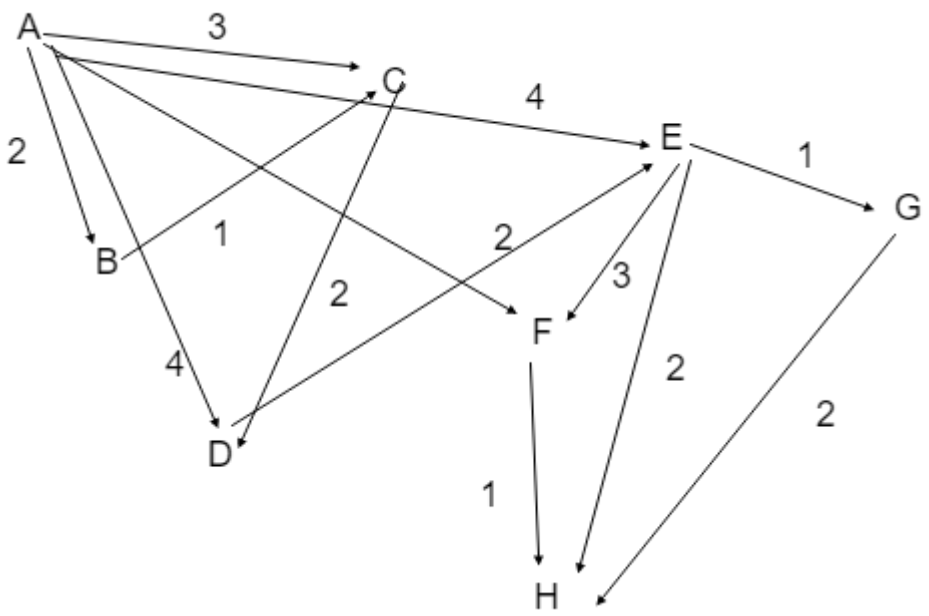
Da har jeg AB = 2, BC = 1, EG = 1, CD = 2, DE = 2, EF = 3, EH = 2
= 13

//Kandidat nr 118. Side 10

11 Oppgave 3b)

Vis hvordan du finner korteste vei fra C til H med Dijkstra's algoritme på grafen fra forrige oppgave. Tegn opp prioritetskøa underveis.

Skriv ditt svar her



C til H:
Den eneste stien fra C er D. D har bare en sti, DE.
E har 3 stier ut fra seg EF, EG, EH. EG er korteset blant de. og blir tatt ut.
Stiene fra F og G til H, er lengre enn EH.

CD(2)
DE(2)
EF(3) EH(2) EH(2)
FH(1) GH(2)

CDEH er da korteste vei

//Kandidat nr 118. Side 11

12 Oppgave 4a)

(Oppgave 4a-c teller 25%) I denne oppgaven kan du benytte tidligere skrevet quadtre kildekode (for eksempel fra Canvas, modul Fasit).

Bruk eller skriv en klasse Quadtre og opprett et quadtre med kun en node (roten). Skriv en funksjon som genererer 20 tilfeldige punkter i området [-2,2] x [-2,2] i xy-planet, altså -2 <= x <=2 og -2 <= y <= 2, og ytterligere 20 tilfeldige punkter i området [0,2] x [0,2]. Punktene skal være av type double eller float. De skal lagres i en array (eller en annen lineær datastruktur) i roten.

Skriv ditt svar her

```
#include <vector>
#include <iostream>
struct Vector2D {
    double x;
    double y;
    Vector2D operator + (const Vector2D& v) const {
        Vector2D u;
        u.x = x + v.x;
        u.y = y + v.y;
        return u;
    }
    Vector2D operator - (const Vector2D& v) const {
        Vector2D u;
        u.x = x - v.x;
        u.y = y - v.y;
        return u;
    }
    Vector2D operator / (int d) {
        Vector2D u;
        u.x = x / d;
        u.y = y / d;
        return u;
    }
};

class QuadTree
{
public:
    QuadTree();
    QuadTree(Vector2D& vne, Vector2D& vnw, Vector2D& vse, Vector2D& vsw,
        QuadTree* qne = nullptr, QuadTree* qnw = nullptr, QuadTree* qse = nullptr,
    QuadTree* Insert(Vector2D input);
    std::vector<Vector2D> data;
private:
    Vector2D v_ne;
    Vector2D v_nw;
    Vector2D v_se;
    Vector2D v_sw;
public:
    QuadTree* q_ne;
    QuadTree* q_nw;
    QuadTree* q_se;
    QuadTree* q_sw;
};

QuadTree::QuadTree()
{
//    //Kandidat nr 118. Side 12
}

QuadTree::QuadTree(Vector2D& vne, Vector2D& vnw, Vector2D& vse, Vector2D& vsw,
    QuadTree* qne, QuadTree* qnw, QuadTree* qse, QuadTree* qsw) :
    v_ne(vne), v_nw(vnw), v_se(vse), v_sw(vsw),
    q_ne(qne), q_nw(qnw), q_se(qse), q_sw(qsw)
{
}

QuadTree* QuadTree::Insert(Vector2D input)
{
//Legger til input i data vektoren
    data.push_back(input);
    return nullptr;
}
void Main(){
    //Danner punktene som den første quaden dekker
```

```
66      Vector2D ne{ 2, 2 };
67      Vector2D nw{ -2, 2 };
68      Vector2D se{ 2, -2 };
69      Vector2D sw{ 2, -2 };
70      //Lager et nytt quadtree, og gir kordinatene til konstruktoren
71      QuadTree* tree = new QuadTree(ne, nw, se, sw);
72
73      //Bruker rand funksjonen modulus grensene til punktene som skal legges inn
74      //For punktene 2 til -2
75      Vector2D va;
76      for (int i = 0; i < 20; i++)
77      {
78          va.x = rand() % 2, -2;
79          va.y = rand() % -2, 2;
80          tree->Insert(va);
81      }
82      //For punktene 0,2 til -0,2
83      for (int i = 0; i < 20; i++)
84      {
85          va.x = rand() % (0,2), -2;
86          va.y = rand() % (-0,2), 2;
87          tree->Insert(va);
88      }
89  }
```

13 Oppgave 4b)

Skriv en funksjon som deler opp de nodene i quadtreet som inneholder flere punkter enn et angitt tall n. Punkter fra hver slik node skal kopieres til de respektive subtrær.

Skriv ditt svar her

```
void QuadTree::DivideOutData()
{
//OBS Denne funksjonen står ikke i deklarasjonen av QuadTree klassen
    //max grensen for hvor mange elementer det kan være i hver quad
    //    //Kandidat nr 118. Side 13
    int max = 5;
    //Finn punkter for hvor mulige nye quads må starte og ende
    //Punkt mellom sør øst og sør west
    Vector2D a = (v_sw + v_se) / 2;
    //Sør øst og nord øst
    Vector2D b = (v_se + v_ne) / 2;
    //Nord øst og nord west
    Vector2D c = (v_ne + v_nw) / 2;
    //Nord vest og sør vest
    Vector2D d = (v_nw + v_sw) / 2;
    //Midten av quaden
    Vector2D m = (v_ne + v_sw) / 2;

    //Ta ut verdier fra data til det er m
    for (int i = data.size()-1; data.size() > max; i--) {

        //if the y is smaller than the middle it should land in one of the south tr
        if (data[i].y < m.y) {
            //If the x is smaller than the middle x, it should land in the left tre
            if (data[i].x < m.x) {
                //Hvis det ikke er noe subtree her, dan en ny med kordinatene funne
                if (q_sw) {
                    q_sw->Insert(data[i]);
                }
                else {
                    //Danner ny quad
                    q_sw = new QuadTree(m, d, a, v_sw);
                    //Setter inn dataen
                    q_sw->Insert(data[i]);
                }
            }
            else {
                if (q_se) {
                    q_se->Insert(data[i]);
                }
                else {
                    q_se = new QuadTree(b, m, v_se, d);
                    q_se->Insert(data[i]);
                }

            }
        }
        else {
            //Opposite of the south
            if (data[i].x < m.x) {
                if (q_nw) {
                    q_nw->Insert(data[i]);
                }
                else {
                    q_nw = new QuadTree(c, v_nw, m, d);
                    q_nw->Insert(data[i]);
                }

            }
            else {
                if (q_ne) {
                    q_ne->Insert(data[i]);
                }
                else {
                    q_ne = new QuadTree(v_ne, c, b, m);
```

```
66                      q_ne->Insert(data[i]);
67                  }
68
69              }
70          }
71          //Remove the element from
72          data.resize(data.size() - 1);
73      }
74      //Kaller del ut igjen, på subtrerne
75      if (q_ne) {
76          q_ne->DivideOutData();
77      }
78      if (q_nw) {
79          q_nw->DivideOutData();
80      }
81      if (q_se) {
82          q_se->DivideOutData();
83      }
84      if (q_sw) {
85          q_sw->DivideOutData();
86      }
87  }
88
89
```

14 Oppgave 4c)

Skriv kode som tester funksjonene fra de forrige to oppgavene og skriver ut punktene for hvert quadtre.

Skriv ditt svar her

```
void QuadTree::PrintAllPoints() {
    //OBS Denne funksjonen står ikke i deklarasjonen av QuadTree klassen
    std::cout << "Mine kordinater: \n";
    std::cout << "NE:(" << v_ne.x << ", " << v_ne.y << "), " << "NW:(" << v_nw.x <<
        << "SE:(" << v_se.x << ", " << v_se.y << "), " << "SW:(" << v_sw.x << ", "
    std::cout << "I have " << data.size() << " elements: " << std::endl;
    for (auto i = 0; i < data.size(); i++) {
        std::cout << "(" << data[i].x << ", " << data[i].y << ")" << std::endl;
    }

    if (q_ne) {
        q_ne->PrintAllPoints();
    }

    if (q_nw) {
        q_nw->PrintAllPoints();
    }
    if (q_se) {
        q_se->PrintAllPoints();
    }
    if (q_sw) {
        q_sw->PrintAllPoints();
    }
}

void main(){
//    //Kandidat nr 118. Side 14
    //Quadtrees
    //Write a quadtree where the blades lay in different levels
    Vector2D ne{ 2, 2 };
    Vector2D nw{ -2, 2 };
    Vector2D se{ 2, -2 };
    Vector2D sw{ 2, -2 };
    QuadTree* tree = new QuadTree(ne, nw, se, sw);

    Vector2D va;
    int minus = 0;
    for (int i = 0; i < 20; i++)
    {
        minus = rand() % 2 ;
        va.x = (rand() % 2) - minus;
        va.y = (rand() % 2) - minus;
        tree->Insert(va);
    }

    for (int i = 0; i < 20; i++)
    {
        va.x = (rand() % (0, 2) - minus);
        va.y = (rand() % (-0, 2) - minus);
        tree->Insert(va);
    }
    tree->Print();
    tree->DivideOutData();

    tree->PrintAllPoints();

    return 0;
}
```