Uppgifter 2

Omfattar: Binära träd, binära sökträd (BST), Heap, Prioritetskö och tidkomplexitet

1. Placera in element med nycklarna 56, 78, 3, 45, 667, 12, 4, 98, 123 i den ordning de är angivna i ett binärt sökträd.

2. Placera in element med nycklarna 56, 78, 3, 45, 667, 12, 4, 98, 123 i den ordning de är angivna i en heap.

3. Beskriv två olika användningsområden för binära träd (olika typer av binära träd).

4. Vad är ett komplett träd binärt träd?

5. Vad är ett balanserat binärt träd?

6. Rita upp ett balanserat binärt träd med 9 noder och höjden 3.

7. Finns det ett komplett binärt träd med höjden 5 som har 37 noder? Motivera.

8. Beskriv kostnaden för att hitta ett element i ett binärt sökträd med n element i bästa fallet och sämsta fallet.

9. Traversera den heap som är representerad av arrayinnehållet: 100, 63, 70, 62, 50, 65, 4, 9, 58 med post-order respektive pre-order.

10. Varför är ett fält lämpligt för att lagra ett komplett träd med n noder?

11. Kan arrayinnehållet

    a. 100, 63, 70, 62, 50, 65, 4, 9, 58
    b. 100, 70, 63, 65, 62, 58, 50, 9, 4
    c. 100, 70, 65, 50, 62, 63, 9, 58, 4

    vara en representation av en heap? Motivera.

12. Hur många jämförelser genomförs som mest vid borttagning från en heap om heapen innehåller 16 element? Motivera.

13. Om innehållet i ett fält är sorterat i omvänd ordning (dvs från största värdet till minsta) är det då en representation av en (max)heap?

14. Gör (rita upp) ett binärt sökträd för elementen 45, 67, 3, 42, 1,2 67, 89, 32, 11, 6. Noderna ska läggas in i trädet i den ordning de är angivna.

15. Gör (rita upp) en heap för elementen 45, 67, 3, 42, 1,2 67, 89, 32, 11, 6. Noderna ska läggas in i trädet i den ordning de är angivna.

16. Antag att Du har en nod-klass som definierar noderna i/för ett binärt sökträd:

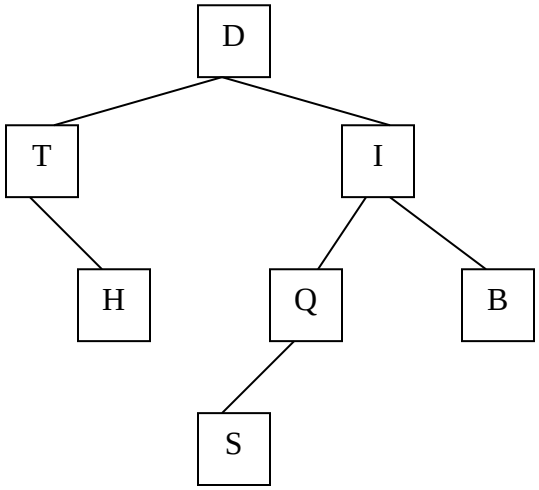
```
template <typename T>
class Node
{
public:
    Node *left;
    Node *right;
    T data;
    Node(T data)
    {
        this->data = data;
        this->left = nullptr;
        this->right = nullptr;
    }
    ~Node(){}
};
```

    Skriv C++-kod för att placera in ett element enligt principen för ett binärt sökträd om parent redan pekar ut den nod som ska bli förälder till den nya noden vars innehåll ska vara info.

17. Antag att ett komplett binärt träd är representerat i ett fält a och att antalet noder i trädet är n. Skriv en funktion som traverserar hela trädet i **inorder** där besöket av en nod innebär en utskrift av nodens innehåll.

18. Visa innehållet i den heap som erhålls om element med följande nycklar placeras i heapen (i den ordning de är angivna): 56, 7, 34, 57, 89, 21, 4, 6, 33, 24, 49. Heapen är representerad med ett fält (array).

19. Antag det binära trädet nedan:



traversera trädet med:

a) preorder
b) postorder
c) inorder

20. Implementera en klass som representerar ett binärt sökträd med ett fält som lagringsstruktur. Det ska vara möjligt att lägga in element, söka efter ett element samt traversera trädet (skriv ut nodernas innehåll). Om Du vill kan Du dessutom tillföra möjligheten att ta bort ett givet element.

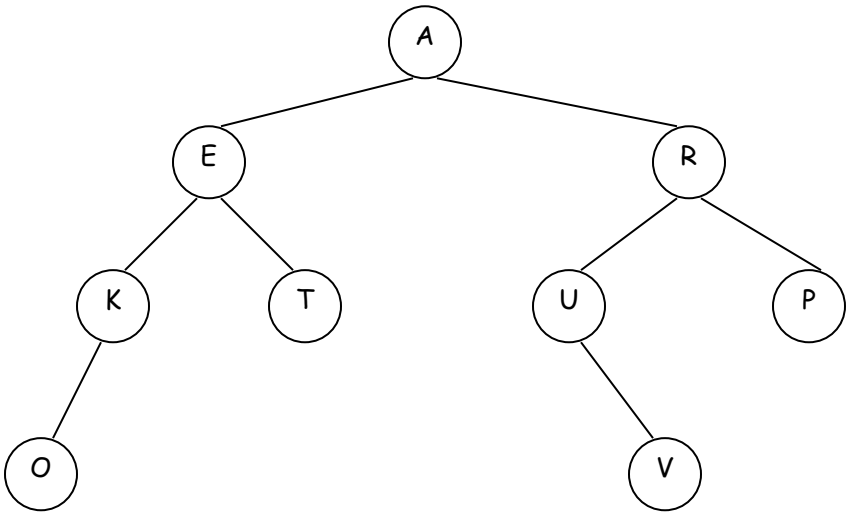
21. Implementera en klass som representerar ett binärt sökträd med länkning som lagringsstruktur. Det ska vara möjligt att lägga in element, söka efter ett element samt traversera trädet (skriv ut nodernas innehåll). Om Du vill kan Du dessutom tillföra möjligheten att ta bort ett givet element.

22. Om ett binärt sökträd traverseras med inorder. Vad kan man då säga om den ordningen vilken nodernas innehåll besöks?

23. Traversera trädet nedan med algoritmen

    d. preorder
    e. postorder
    f. inorder

    Ange för de olika algoritmerna i vilken ordning noderna besöks.



24. Hur kan man gå till väga för att lägga upp ett balanserat binärt sökträd om det som ska placeras i trädet är sorterat i ett fält (en array) av storlek n?

25. Skriv implementation för att ta bort det minsta värdet/elementet i ett binärt sökträd om noderna i trädet representeras av klassen Node nedan samt att pekaren root pekar på rotnoden i det binära sökträdet:

```
template <class T>
class Node
{
public:
    Node *left;
    Node *right;
    T data;
    Node(T data)
    {
        this->data = data;
        this->left = nullptr;
        this->right = nullptr;
    }
    ~Node(){}
};
```

26. Skriv implementation för att traversera alla noder i ett komplett binärt träd om trädet är representerat med en array deklarerad T *tree och där du dessutom har tillgång till heltalsvariabeln nrOfNodes som innehåller antalet noder som finns i trädet.

27. Hur används enkellänkning lämpligen vid implementation av en Prioritetskö om tiden för operationerna dequeue ska vara konstant O(1) samt tiden för operationen enqueue ska vara O(n) där n representerar antalet element?
28. Hur används en array lämpligen vid implementation av en Prioritetskö om tiden för operationerna dequeue ska vara O(n) samt tiden för operationen enqueue ska vara O(1) där n representerar antalet element?

29. Hur används en array lämpligen vid implementation av en Prioritetskö om tiden för operationerna dequeue och enqueu ska vara O(nlogn) där n representerar antalet element?

30. Betrakta följande kodavsnitt

```
for (int i=1; i<=n; i++)
   a[i] = 0;
for (int i=1; i<=n; i++)
   for (int j=0; j<n; j++)
      a[j] = a[i] + a[j] + i*j;
```

a) Vilken är tidskomplexiteten T, och den asymptotiska tidskomplexiteten O för kodavsnittet ovan?
b) Om det tar 1 sekund att exekvera kodavsnittet ovan om n=1000, hur lång tid tar det då om n ökas till 10000? Samma förutsättningar gäller i båda fallen och O-notationen är tillräcklig.

31. Betrakta följande funktion

```
int spam(int n)
{
    int result = 0;
    for (int i=1; i<n; i++)
       for (int j=0; j<n*n; j++)
          result += i*j;
    return result;
}
```

a) Vilken är tidskomplexiteten T, och den asymptotiska tidskomplexiteten O för kodavsnittet ovan?
b) Om det tar 10 sekund att exekvera kodavsnittet ovan om n=100, hur lång tid tar det då om n ökas till 5000? Samma förutsättningar gäller i båda fallen och O-notationen är tillräcklig.

32. Betrakta följande funktion

```
int spam(int n)
{
   int result = 0;
   int i = 0;
   while (i<n)
   {
      for (int j=1; j<n; j++)
        result += i*j;
      i++;
   }
   return result;
}
```

a) Vilken är tidskomplexiteten T, och den asymptotiska tidskomplexiteten O för kodavsnittet ovan?
b) Om det tar 30 sekund att exekvera kodavsnittet ovan om n=100, ungefär hur stort n klarar samma dator på 2 minuter? Samma förutsättningar gäller i båda fallen och O-notationen är tillräcklig.