TENTAMEN 20150112 DVA104 LÖSNINGSFÖRSLAG

UPPGIFT 1

a) Utan ett basfall i en rekursiv funktion kan inte funktionen avslutas utan att programmet krashar.

b) En algoritm är stabil om den bibehåller den relativa ordningen mellan två (eller flera) element med samma nyckel/data.

c) LIFO-kö är ett annat namn på en stack (Last In First Out)

d) Att en algoritm har konstant komplexitet innebär att körtiden/effektiviteten inte är beroende av datamängden. Effektiviteten är alltså densamma oavsett storleken på mängden. O(1)

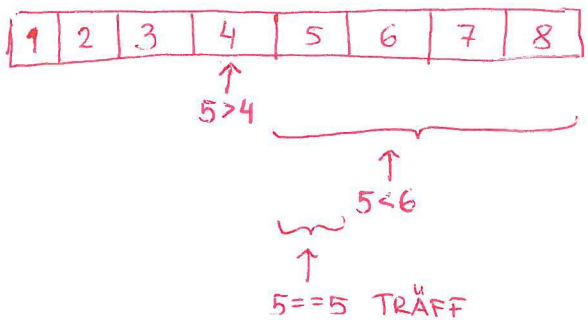
e) Man kan optimera standardlösningen av en bubblesort genom att
1. Inte jämföra med de data som vi kan vara säkra på redan fått sin rätta plats. För varje runda/varv får ett tal (största eller minsta beroende på implementation) sin rätta plats.
2. Avbryta algoritmen om ett helt varv körts utan några byten, det betyder att samtliga element ligger på rätt plats.

UPPGIFT 2

a) Vid binärsökning i en linjär sekvens jämför man det mittersta värdet i sekvensen med det eftersökta. Får man träff är sökningen färdig, är det eftersökta mindre utförs binärsökning i vänster delsekvens och är det eftersökta större utförs binärsökning i höger delsekvens.

För att binärsökning ska kunna genomföras måste sekvensen vara sorterad.

ex. sök efter 5 i följande sekvens



TENTAMEN 20150112 DVA104
LÖSNINGSFÖRSLAG

UPPGIFT 2 FORTS.

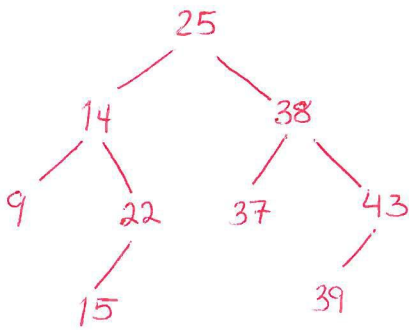
b)

```
int binarySearch (int arr[], int searchFor, int low, int high)
{
    int mid;
    if (low <= high)
    {
        mid = (low+high)/2;
        if (arr[mid] == searchFor)
            return 1; //Found
        else if (searchFor < arr[mid])
            return binarySearch(arr, searchFor, low, mid-1);
        else
            return binarySearch(arr, searchFor, mid+1, high);
    }
    return 0; //not found
```

c) Komplexitet i värsta fallet (worst case) är O(log n) eftersom algoritmen vid varje varv i sökningen halverar sökmängden.

UPPGIFT 3

a) 25 14 38 43 9 39 22 15 37



Binärt = varje nod har max 2 barn

Sorterad = sökträd = mindre värden till vänster och större värden till höger i varje deltråd.

b) preorder = behandla datat, gå vänster, gå höger.

25 14 9 22 15 38 37 43 39

c) Ja, trädet är balanserat, det går inte att bygga om trädet för att få färre nivåer.

Faktiskt djup = 4

Teoretiskt djup = (int) $\log_2(9)+1 = 4$ (9 = antal noder)

UPPGIFT 3 FORTS.

d)
```
typedef struct Node{
        int data;
        struct Node *left;
        struct Node *right;
} Node;
```

e)
```
Node *insertSorted (Node *subTree, int data)
{
    if (subTree == NULL)
    {
        Node *newNode = (Node *)malloc (sizeof(Node));
        if (newNode != NULL)
        {
            newNode -> data = data;
            newNode -> left = NULL;
            newNode -> right = NULL;
            subTree = newNode;
        }
        else
            printf ("error");
    }
    else
    {
        if (data < subTree -> data)
            subTree -> left = insertSorted (subTree -> left, data);
        else
            subTree -> right = insertSorted (subTree -> right, data);
    }
    return subTree;
}
```

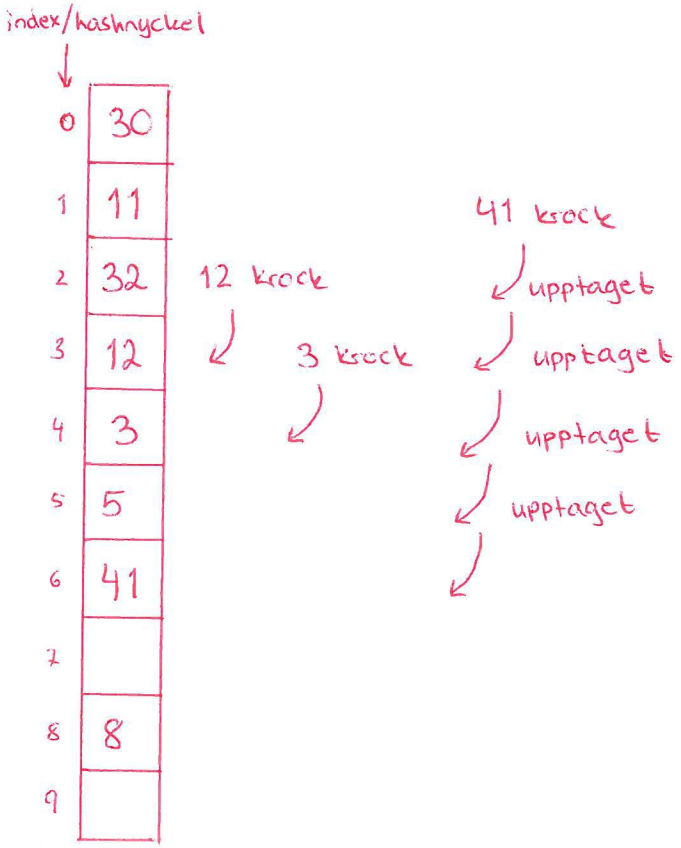
f) Vi måste hitta på djupet i samtliga deltråd för att kunna avgöra vilket som är djupast. Vi kan inte veta hur många deltråd som finns i det binära trädet.
Binära träd är därför naturligt rekursiva.

TENTAMEN 20150112 DVA104
LÖSNINGSFÖRSLAG

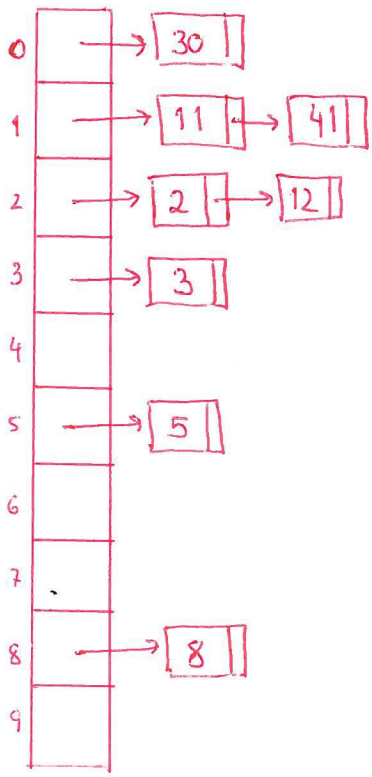
UPPGIFT 4

x:	32	5	12	8	3	11	30	41
x%10:	2	5	2	8	3	1	0	1

a)



b)

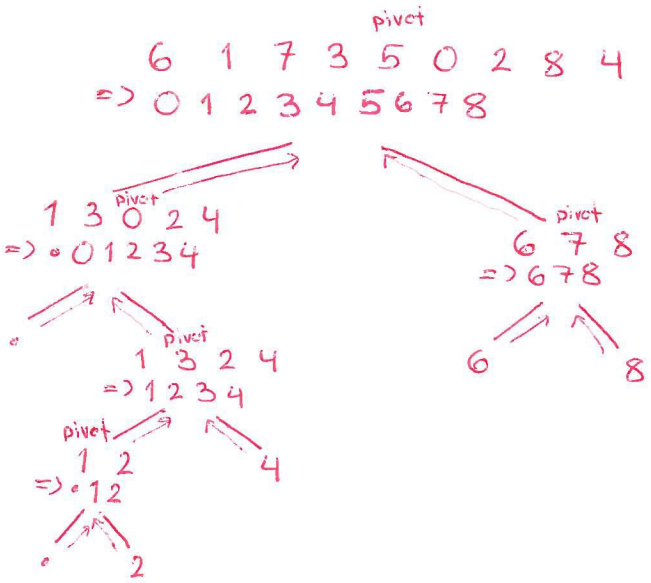


c) Onödigt mycket minne går åt, man allokerar platser som inte kommer användas. Beroende på data och hashfunktion så är det inte heller säkert att man lyckas undvika/minska mängden krockar, i en länkad hashtabell leder varje krock till att ytterligare platser i hashen inte används.

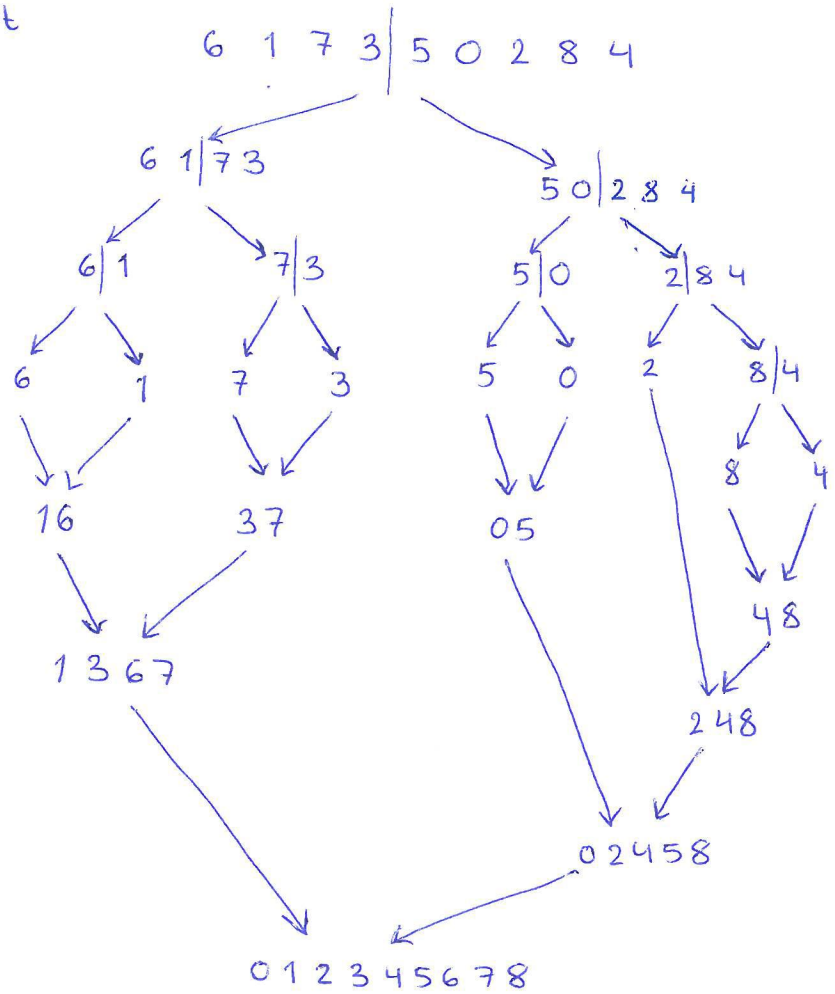
TENTAMEN 20150112 DVA104
LÖSNINGSFÖRSLAG

UPPGIFT 5

a) Quicksort



MergeSort



TENTAMEN 20150112 DVA104
LÖSNINGSFÖRSLAG

UPPGIFT 5 FORTS.

b) Väljs ett dåligt pivot kan algoritmen degenerera, alltså gå mot $O(n^2)$.
Ett bra pivot delar hela tiden mängden på hälften men att beräkna vilket som är ett optimalt pivet för varje runda är kostsamt ($O(n^2)$). I medeltal har algoritmen $O(n \log n)$ men då finns risk att den degenererar.
Att välja första eller sista elementet till pivot i en sorterad eller bakåtsorterad sekvens ger en degenererad algoritm.

UPPGIFT 6

```
MergeSort(integer first, integer last)
    integer mid
    if first < last
        mid = (first + last)/2
        MergeSort(first, mid)
        MergeSort(mid+1, last)
        Merge(first, mid, last)
```