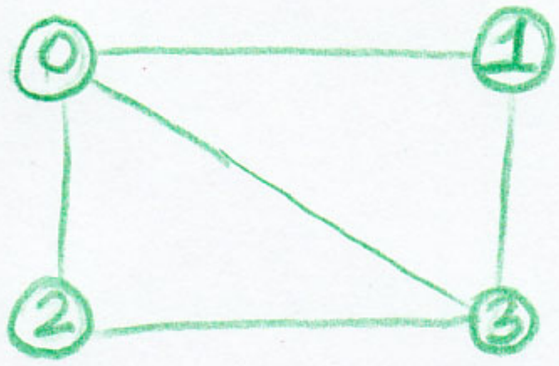
A

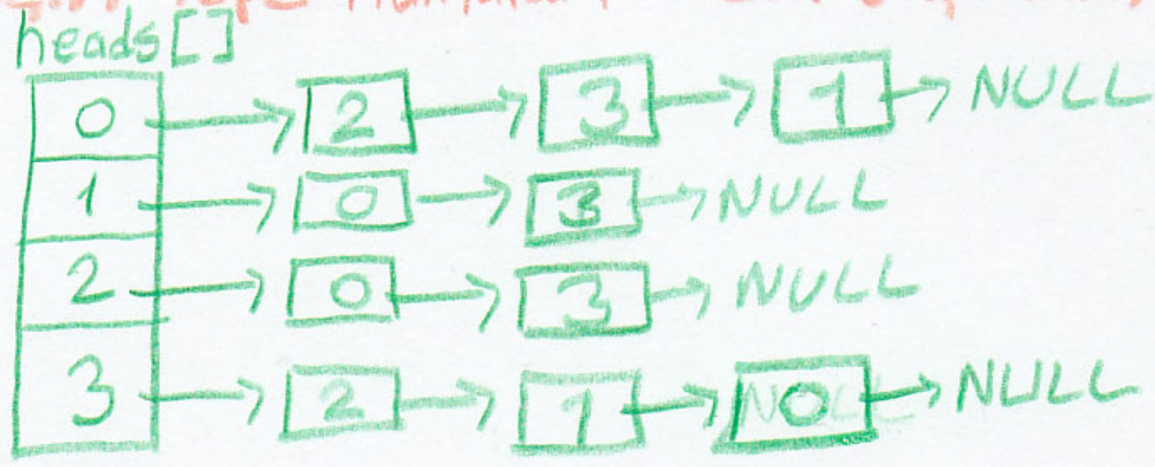
CME227 DATA STRUCTURES
Final, December, 29, 2014

Number/Name/Division:..... Duration: 75 minutes

1-) (20 pts.) Write a function that returns true if there is an edge between given two vertices in an undirected graph which is represented by an adjacency list, false otherwise.

Ne istendiğini anlamak için önce basit bir graf çizip bunu komşuluk listesi ile gösterelim.
C ile uyumlu olması için tepe numaları 0'dan başlasın,





Grafta örneğin 0 ile (3) numaralı tepeler arasında ayrıt olup olmadığını belirlemek için heads[0] listesinde datası (3) olan düğümü arayacağız. Bulursak true döndüreceğiz.

```
int edge(struct node *heads[], int u, int v){
  struct node *temp = heads[u];
  while (temp->next != NULL){
    if(temp->next->data == v)
      return 1;
    temp = temp->next;
  }
  return 0; // bulunmadı
}
```

sizden istenen sadece 7 satırlık kod idi!

2-) (20 pts.) What is the smallest and largest number of nodes in a heap of height 8? What is the height of a heap with 600 nodes? **Height of a binary tree: The number of edges on the path from the root to its deepest descendant.**

Bir heap son seviyesi hariç tüm seviyeleri dolu olmak zorunda olan bir ikili ağaçtır. Son seviye full olabilir.

level 8 de 1 düğüm olursa minimum düğüm sayısını elde ederiz.
Herhangi bir level'daki düğüm sayısı 2^{level} dir. (5. soruda var)

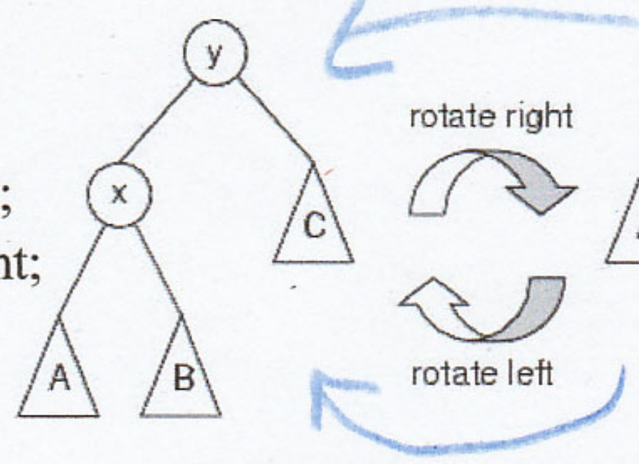
$min = 2^0 + 2^1 + 2^2 + \dots + 2^7 + 1 = 2^8 = 256$

$max = 2^0 + 2^1 + \dots + 2^7 + 2^8 = 2^9 - 1 = 511$

600 düğümlü bir heap'in yüksekliği 9 olur. Geriye kalan 89 düğüm level 9 a sığar.

3-) (20 pts.) We suppose that an AVL tree node has the following data structure.

```
struct node{
    int key;
    struct node *left;
    struct node *right;
    int height;
};
```




```
struct node *leftRotate(struct node *x){
    ........
    ........
    ........
    return .....
}
```

Complete the above function that performs left rotation. Note that height member of some nodes should be updated.

```
struct node *leftRotate(struct node **){
    struct node *y = x->right;
    {x->right = y->left;
    {y->left = x;
    {x->height = max(x->left->height, x->right->height)+1;
    {y->height = "  y  "  y  "
    return y;
}
```

Burada sıra önemli önce aşağılar yapılmalı

4-) (20 pts) Write a function that adds the maximum number of a binary search tree to all nodes in the tree. It is not allowed to use functions defined in the class.

```
int max ( BTREE root)
    if(root!=NULL){
        while (root->right!=NULL)
            root=root->right;
        return root->data;
        }
}
void addMax (BTREE root, int max.){
    If (root!=NULL){
        addMax(root->left, max);
        root->data+=max;
        addMax(root->right, max);
        }
}
/* ilk cagirma */
addMax( root, max(root));
```

5-) (22.5 pts.) Circle and fill T or F for each of the following statements to indicate whether the statement is **true** or **false**, respectively. If the statement is wrong, explain why.

- T **F** The possible maximum number of edges in a graph with n nodes is n^3.
- T **F** The heights of any two siblings in a binary heap differ by at least 1.
- **T** F Any subtree of an AVL tree is always itself an AVL tree
- T **F** An AVL tree is always the same after deletion and then insertion the same node
- T **F** A tree is a graph that may have cycle
- **T** F Adjacency list representation of graphs is less efficient than adjacency matrix representation for dense graphs.
- T **F** Recursive function calls use queue
- T **F** A binary heap is also an AVL tree
- **T** F The length of a path is the number of edges on that path
- **T** F The possible maximum number of nodes on level k of a binary tree is $\mathbf{2^k}$
- T **F** Depth of a node in a tree is the number of edges on the path from the node to the deepest leaf
- T **F** Every node in a tree has at most 2 children.
- T **F** Linked list nodes are normally stored contiguously in memory.
- T **F** A stack is a linked-list that can be accessed from either end.
- **T** F Push is used to place elements on the bottom of a stack and pop is used to remove elements from the top of a stack.