Week 6 Solutions:

1. **A sparse graph is a graph in which the number of edges in very less. Which of following representation will be better to represent a sparse graph?**

    A. **Adjacency list representation**
    B. **Adjacency matrix**

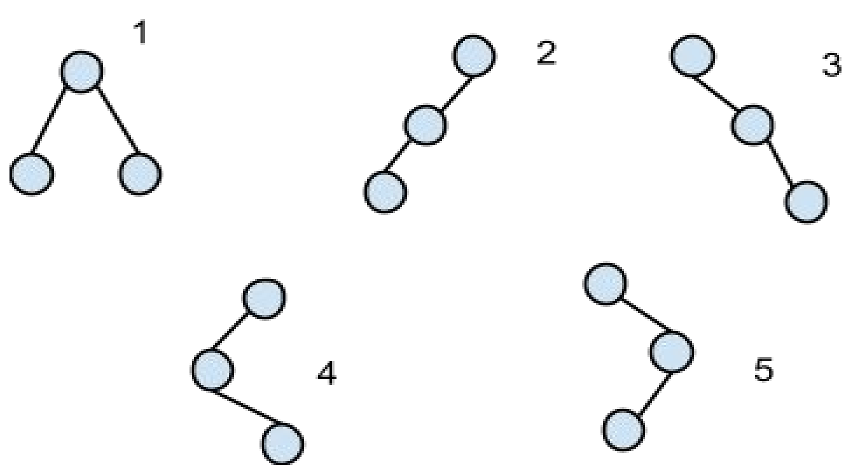
Answer: A

If you store the graph as a matrix, most of the entries in the matrix will be zero. While the space required to store as adjacency list representation will be less.

2. **The maximum number of binary trees that can be formed by 3 nodes is:**

A.3
B.9
C.1
D.5

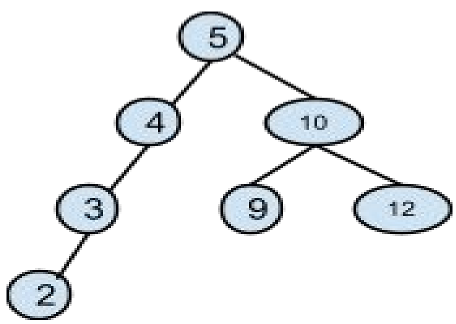
Answer: D



3. **Insert 5,4,10,3,9,2,12 in an empty binary search tree (BST) in the sequence, Which is the element will be in the lowest level ?**

A.2
B.5
C.12
D.3

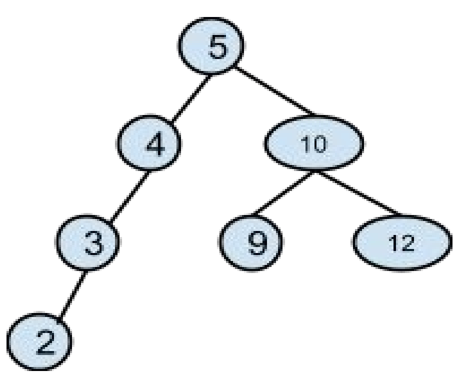
Answer: A



4. **Preorder traversal of a BST is 5,4,3,2,10,9,12. What will be the postorder traversal of the tree?**
    - **A. 2,3,4,5,9,10,12**
    - **B. 2,3,4,10,12,9,5**
    - **C. 2,3,4,9,12,10,5**
    - **D. Cannot be determined From data given in Question**

Answer: C

Since it's given that tree is BST, inorder will be 2,3,4,5,9,10,12 So tree can be uniquely constructed as



5. **Which traversal is the most suitable for deleting all the nodes in a binary tree ? (You can assume the tree to be binary)**
    A. **Inorder**
    B. **Preorder**
    C. **Postorder**
    D. **Any traversal**

    Answer: C

    Before deleting the parent node we should delete its children first

6. **Identify what *func* does ?**

```
struct BTNode{

        struct BTNode * left;
        struct BTNode * right;
        int value;
};

int func(struct BTNode * root) {

        int i_left, i_right;

        if(root == NULL)
           return 0;
        else{

                i_left   = func(root->left);
                i_right = func(root-> right);

                if(i_left > i_right)
                      return (i_left+1);
                else
                      return (i_right+1);
        }
}
```

    A. **Find number of nodes in the tree**
    B. **Find depth of the tree**
    C. **Find number of leaves in the tree**
    D. **None of the above**

    Answer: B

    Finding the height of the left and right subtree of a node and assign height to the node as maximum of (left and right subtree + 1)

7. **This function *SumElements* finds the sum of all the elements in the binary tree. Complete the code**

```
struct BTNode{
        struct BTNode * left;
        struct BTNode * right;
        int value;
};

int SumElements(struct BTNode * root){

   if(root == NULL)
       return 0;
  else
     return ________
}
```

   A. root->data + SumElements(root->left)
   B. root->data + SumElements(root->right)
   C. root->data + SumElements(root->left) + SumElements(root->right)
   D. SumElements(root->left) + SumElements(root->right)

   Answer: C
   Need to add recursively, left subtree sum, right subtree sum and add value of current node

8. **The given code checks whether the given binary tree is a Binary search tree or not. Assume that we have *FindMax* and *FindMin* that return the min or max integer value from a non-empty tree.**

```
struct BTNode{

        struct BTNode * left;
        struct BTNode * right;
        int value;
}

int isBST(struct BTNode * root){

         if(root == NULL)
            return 1;
         if(root->left !=NULL && FindMin(root->left >    root->data) //P
           return 0;
         if(root->right !=NULL && FindMax(root->right  < root->data)// Q
           return 0;
        if(!isBST(root->left) || !isBST(root->right)
            return 0;
        return 1;
}
```

A. No Logical error
B. Error in line "P"
C. Error in line "Q"
D. Error in both lines "P" and "Q"

Answer: D
Checking at the current node is not sufficient.
Rather we need to check if max value in left subtree is smaller than the current node and minimum value in the right subtree is larger than the node data
So, in

P : it should be FindMax
and
Q: it should be FindMin

9. For a binary heap of height h, what are the minimum and maximum number of elements in the heap.

A. 2^h -1 and 2^(h+1) -1
B. 2^h and 2^(h+1) -1
C. 2^h and 2^(h+1)
D. 2^h -1 and 2^(h+1)

Answer: B
Number of nodes will be minimum if the last level has just one node. Since it is a heap the previous level must be completely filled (2^h-1 + 1) = 2^h nodes
Number of nodes will be maximum when the last level is completely filled.
Thus 2^(h+1)-1

10. Suppose there are k sorted lists (decreasing order) with n/k elements in each list. What is the time complexity to merge them into one single sorted list.

Hint: Maintain a heap of k elements. Think which k elements to choose.

A. O(nlogk)
B. O(n)
C. O(nk)
D. O(nlogn)

Answer: A
Algorithm will be:

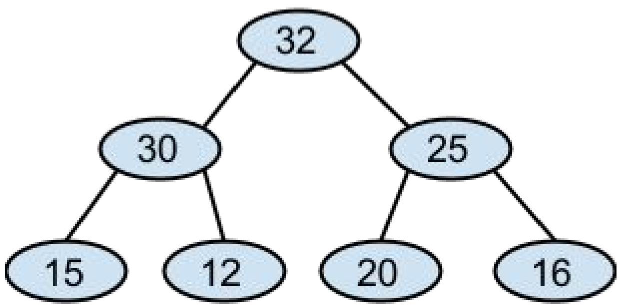
1. Build the max-heap with all first elements from each list
2. In each step extract the maximum element and add at the end
3. Add next element from the list of the one extracted
4. Repeat step-2 and step-3 until all elements are completed from the list
   k elements max-heap and for all elements we need to update heap in l*ogk* time

Hence

$O(nlogk)$

11. **The elements 32,15,20,30,12,25,16 are inserted one by one in the given order into a MAX heap that is initially empty. Which of the following node does not lie in the right subtree of the root ?**
    - **A. 12**
    - **B. 25**
    - **C. 16**
    - **D. 20**

Answer: A



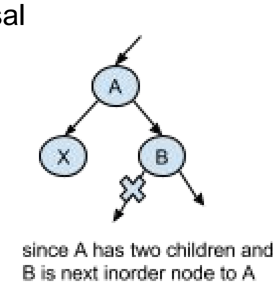
12. Consider a Binary Tree. Let A and B be two nodes in the tree. It is given that B is the inorder successor of A, and A has 2 children. Which of the following is true about B?

Hint: Successor means the next node

- **A. B has no right child**
- **B. B has no left child**
- **C. B has both children**
- **D. None of the above**

Answer: B

B should be the left most node in the right subtree to be the successor in inorder traversal



13. Following is C like pseudo code of a function that takes a stack *s* and an element *el*, determine the status of the stack S passed after the call to *DoSomething*().

```
void DoSomething (Stack s, int el) {
   Stack tmp;  // Say it creates an temporary empty stack
   while (!s.isEmpty()){
                  tmp.push(s.pop())
   }
          s.push(el);
          s = tmp;
}
```

A. *s* will remain same
B. *s* will contain only *el*
C. *s* will contain *el* followed by previous elements of stack in same order
D. *s* will contain *el* followed by previous elements of stack in reverse order

Answer: **B**

In the code all the element of *s* is being removed and an element *el* is pushed.
The statement *s* = *tmp*, will only bind *s* with value of tmp and will not change the value, hence stack will only contain the new el

14. Which one of the following is an application of Stack and Queue Data Structure respectively?

A. function calls and arithmetic expression evaluation
B. resource sharing and stock span problem
C. function calls and stock span problem
D. tower of hanoi and printer spooler

Answer: D
Applications of stack : function calls, arithmetic expression evaluation, stock span problem, tower of hanoi
Applications of queues : resource sharing, printer spooler
Only option D have stack and queue operation ordered correctly.

15. Consider the following pseudo code. Assume that IntStack is an integer stack. What does the function DoSomething do?

```
void DoSomething(int n)
{
   IntStack s = new IntStack();
   s.push(0);
   s.push(1);
   for (int i = 0; i < n; i++)
   {
      int a = s.pop();
      int b = s.pop();
      s.push(a);
      s.push(a + b);
      printf("%d",b);
   }
}
```

- **A. Prints numbers from 0 to n-1**
- **B. Prints numbers from n-1 to 0**
- **C. Prints first n Fibonacci numbers**
- **D. Prints first n Fibonacci numbers in reverse order.**

Answer: C

Everytime top of the stack is pushed with next fibonacci number which is being printed

16. Assume that the operators +, -, × are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, x , +, -. Find the postfix expression corresponding to the infix expression

a + b × c - d ^ e ^ f

- **A. abc × + def ^ ^ -**
- **B. abc × + de ^ f ^ -**
- **C. ab + c × d - e ^ f ^**
- **D. - + a × bc ^ ^ def**

Answer: A

Putting rounding braces based on precedence and associativity can be given by

((a + (b × c)) - (d ^ (e ^ f)))

Hence postfix expressions will be given by abc × + def ^ ^ -