```
/***********************************************************************************
**                                                                               **
**      Name:           Clyde Pabro                                              **
**      Class:          CISP 430 – Fall 2012 Thu                                 **
**      Assignment:     Mid Term                                                 **
**                                                                               **
***********************************************************************************/
```

I declare that the following exam is my own work, with the exception of those portions which are properly documented.

Signed:________________________________________Date:___________________________________

## ANSWER 1:

The concept of time complexity is the general framework to analyze algorithm efficiency. It is important because it a powerful technique to analyze the efficiency of algorithm's runtime with mathematical precision. The code analysis of time complexity investigates the algorithm's efficiency as a function of some parameter n indicating the algorithm's input size or its rate of growth. The mathematical techniques are comparing run-time with constant (1), linear (n), logarithmic ( logn), linear logarithmic (nlogn), quadratic (n^2), cubic (n^3), and exponential (2^n). The empirical techiques are  the Big-O notations with best case and worst case scenarios.

The concept of Big-O notation is the upper bound of a given function with its rate of growth. It uses an asymptotic analysis to determine the running time of an algorithm.  The code analyzed are loops, nested loops, consecutive statements, if-then-else statements, and logarithmic complexity. The mathematical techniques are O(1), O(n), O(n^2), O(n^3), O(log n), O(nlogn), and O(n/2). The empirical techniques are best case and worst case scenarios for the rate of growth of a given function.

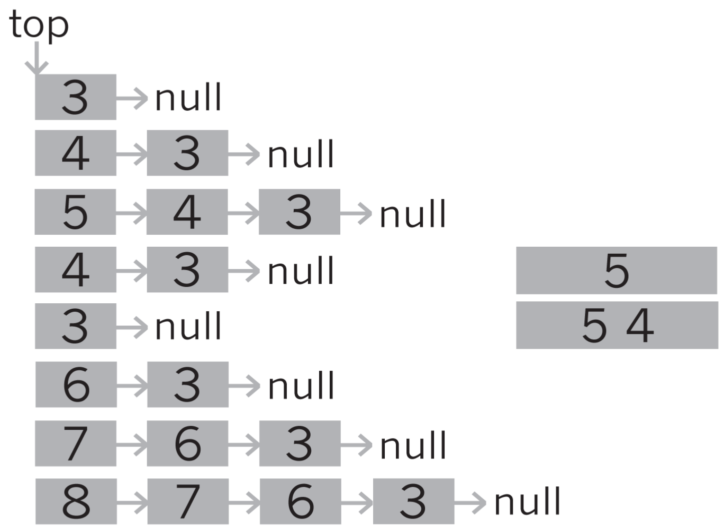
## ANSWER 2:

a. O( n^2 ) because the For loop is O( n ) so its loop goes n times. For the embedded For loop goes n times, so the time complexity of the entire algorithm is O( n ) x O( n ) = O( n^2 ).
b. O( n ) because it traverses through the list n times.
c. O ( log n) because it has to cycle through the list with a while loop to search through the trees. The worst case  for a non-fully filled tree having only one branch is O(n ).
d. O( 1 ) because the pop() function has no loops and is a constant time complexity.

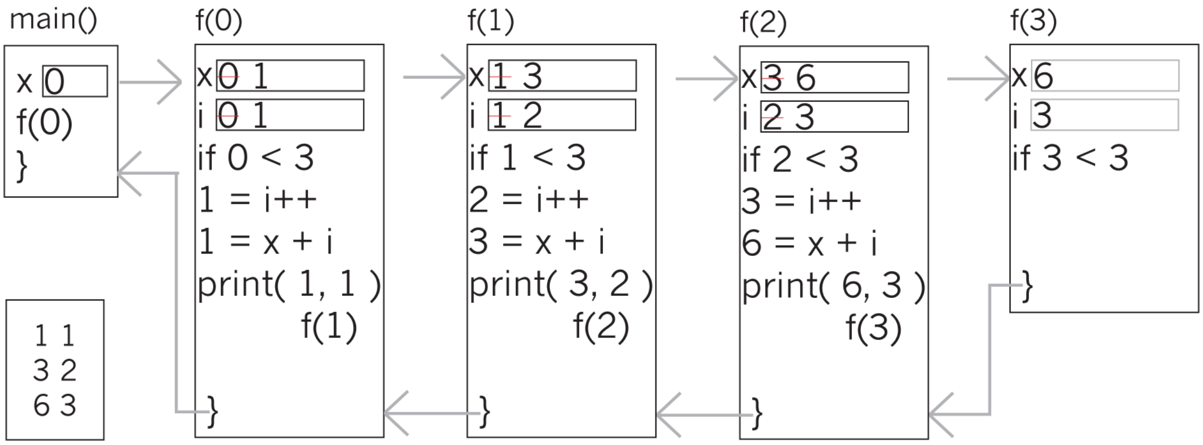
## ANSWER 3:

```
initstack(void)
{
        *head = 0;
        *tail = 0;
}

void push( item entry )
{
        node * temp = (node*)malloc(sizeof(node));
        temp->entry = entry;
        temp->next = head;
        head = temp;
}

item pop( void )
{
        data d;
        node *temp;
        if( isEmpty() )
                return 0;
        temp = head;
        head = head->next;
        d = temp->d;
        free(temp);
        return d;
}

bool isEmpty()
{
        if( head )
                return false;
        else
                return true;
}
```

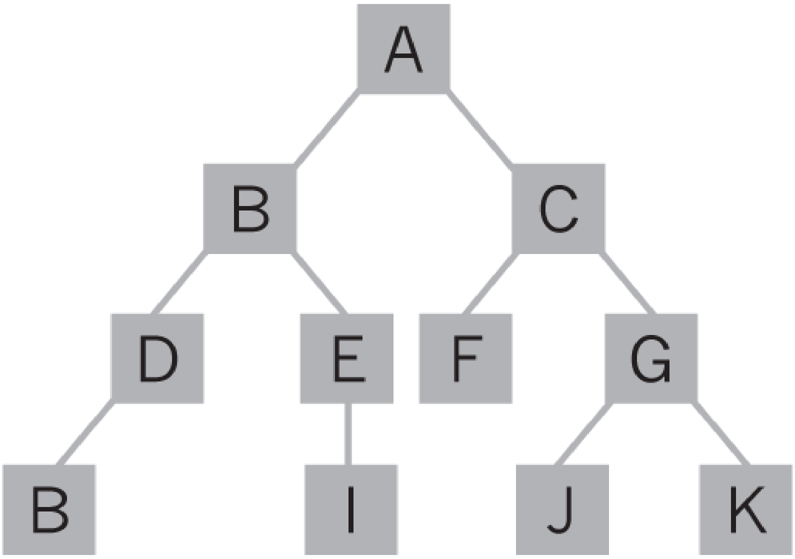
**ANSWER 4:**



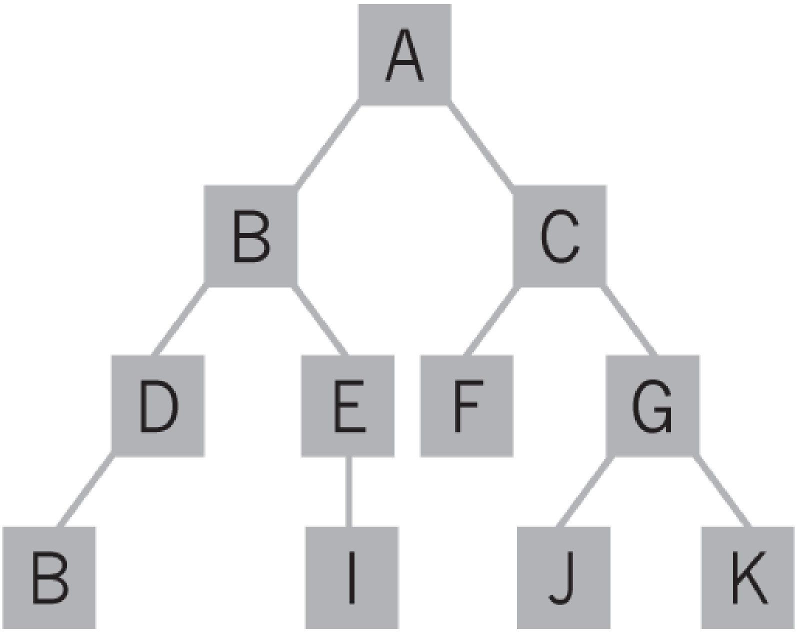
**ANSWER 5:**



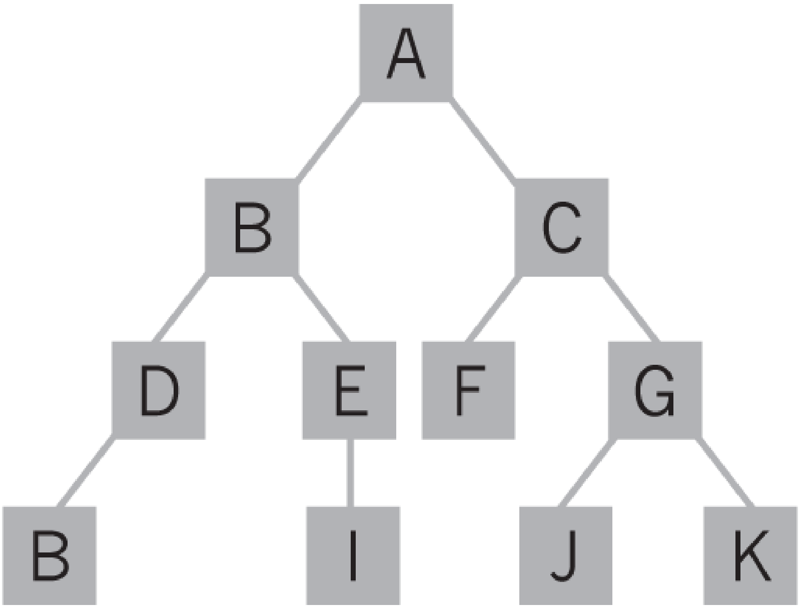
**ANSWER 6:**



traverseNLR: A H D I E B K G J C F



traverseLNR: H D B I E A K J G  F C



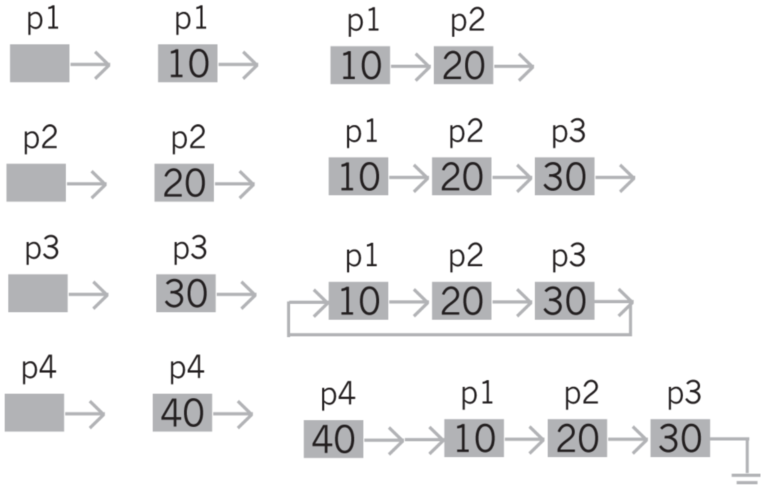
traverseLRN: H D I E B F J K G C A

**ANSWER 7:**

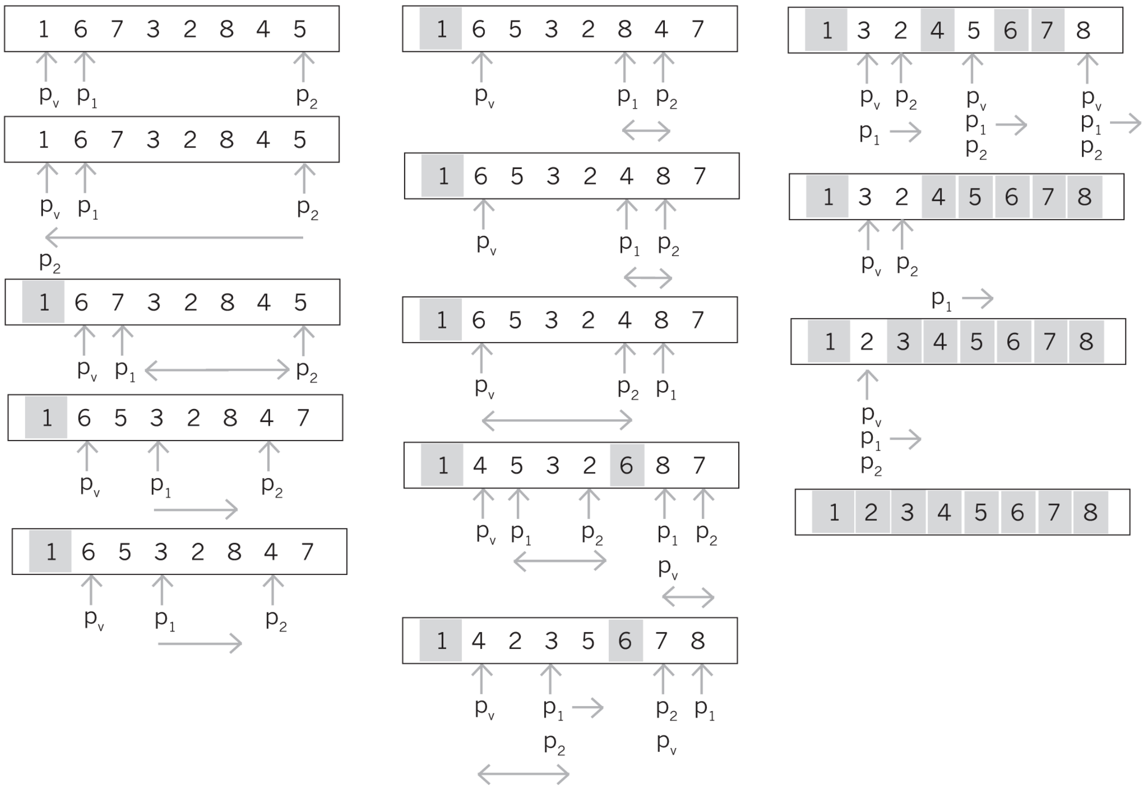
```
void traverse( node * root ) {
        if( root != NULL ) {
                int ans
                process( root );
                traverse( root->left );
                traverse( root->right );
                if( find() )
                        cout << data << endl;
        }
}
int find( node * root, data d ) {
        int temp;
        if( root == NULL ) {
                return 0;
        } else {
                temp = find( root->left, data )
                if( temp != 0 )
                        return temp;
                else
                        return( find( root->right, int data )
        }
        return 0;
}
```

The time complexity of the algorithm is O( n ).

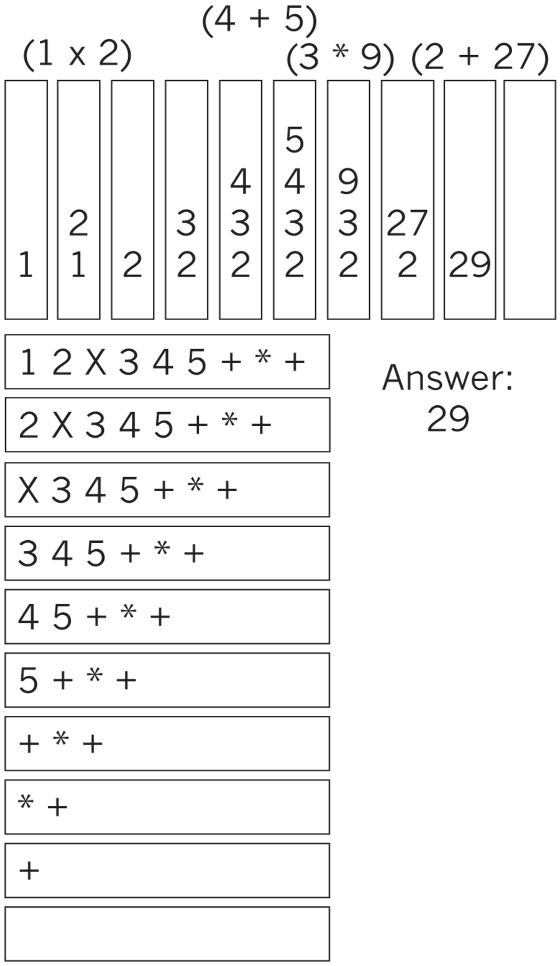
**ANSWER 8:**



**ANSWER 9:**



**ANSWER 10:**



Answer:
29