Easy

1. Name 5 stages of a program life cycle?
2. f(n)=4n3+6n2+7n , what is the algorithmic complexity of f(n) in O (Big Oh) notation
3. What is the memory address of y[2][2][3] if the memory address of y[0][0][0] α and y is declared as y[5][4][4]?
4. Trace the given recursive call as we did in class

```
int sum( int arr[], int n )
{
        if ( n == 0 )
                return 0;
        else
        {
                int smallResult = sum( arr + 1, n - 1 ); // "A"
                return smallResult + arr[ 0 ];
        }
}
```

Medium

1. Trace the given recursive call as we did in class and find what RecursiveFunc do in general?

```
int RecursiveFunc (int array[], int index, int n)
{
        int val1, val2;
        if ( n==1 )
                return array[index];
        val1 = RecursiveFunc (array, index, n/2);
        val2 = RecursiveFunc (array, index+(n/2), n-(n/2));
        if (val1 > val2)
                return val1;
        else
                return val2;
}

.. . .
int a[7] = {1,2,10,15,16,4,8};
printf( " value is %d\n", RecursiveFunc(a,0,7));
```

Solution: 16 - function finds the maximum

2. Write a recursive version of the below iterative solution of Fibonacci sequence code? Compare iterative and recursive version. Give two advantages for both.

```
int Fib(int n)
{

   int prev1, prev2, tem, j;

   if(n==0 || n==1)
      return n;
   else{
       prev1=0;
       prev2=1;
       for(j=1;j<=n;j++)
       {
           temp = prev1+prev2;
           prev1=prev2;
           prev2=temp;
       }
       return prev2;
   }
}
```

3. What is the order of growth of the worst-case running time of each of operation below?

2-Sum	
3-Sum	
Iterative Fibonacci	
Binary Search	

Solution: N^2 , n^3, n, logn

4. Approximately how many array accesses as a function of input size N?
Give some code here similar to slides

5. Given an array A[2][3] and the main memory represented as M[],
   In row major ordering what is the index of A[1][2] in M?
   In column major ordering what is the index of A[1][2] in M?

6. Write the output of below code segment? Assume the address of z[0][0] is 100.

```
int z[4][2] = { {2,4}, {6,8}, {1,3}, {5,7}};
printf("z[0] = %p\n", z[0]);
printf("  *z = %p\n", *z);
printf("   z = %p\n", z);
printf("   z[2][1] = %d\n", z[2][1]);
```

7. For the same above question, what is the memory address of z + 1 ?

8. A(x) and B(x) are two polynomials defined as
   A(x) = 3x ^20 + 2x^ 3 + 4 B(x) = x^5 +10x ^3 +3x ^2 +1
   In class we have shown a better structure to store polynomials.
   Ask to compare the analysis of addition operation and storage for new and old structure.

9. Assume a new type of special matrix, named stripe matrix, that has non-zero values in odd numbered columns as shown in below.
   a) Design a new structure to hold elements of the below matrix and describe the storage benefit of your newly designed structure compared to a traditional double array storage.
   b) Give the number of elements as a function of matrix rox/column size N.
   c) Give the location of a given matrix element u[i][j] as a function of matrix rox/column size N.

   (For simplicity, assume all stripe matrices are N by N matrices.)

2	0	3	0
4	0	3	0
5	0	-3	0
20	0	12	0

10 . Assume a new type of special matrix, named stripe matrix, which has non-zero values in every a^{th} column. Below is a NxN stripe matrix M(N, a) where a is 3.

For this matrix, we would like to store the values, and their coordinates of the values to save space on the main memory.

a) Give the space usage of such representation as a function of N and a.

b) Explain under what circumstances such representation will use less space than traditional double array storage.

0	0	2	0	0	3	0
0	0	4	0	0	3	0
0	0	5	0	0	-3	0
0	0	20	0	0	12	0

Solution : a > 2 will save space , since we save 3 values to store 1 value, so at most we should have n/3 items in the array to save space.

Hard

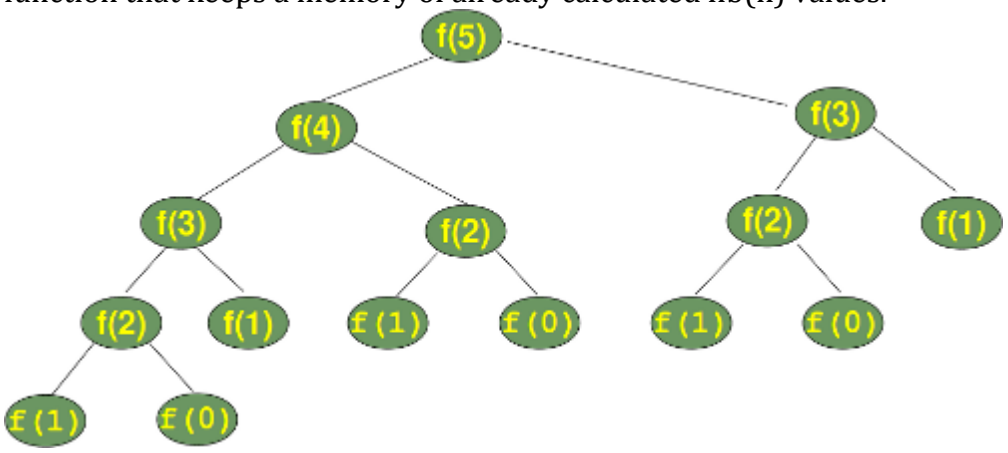
1. Explain the output and end result of below recursive call.

```
call(int n)
{
        print(n)
        call (n+1)
}
```

Solution : Recursive tracing goes infinitely. Stack overflow.

2. The problem with the recursive Fibonacci number solution was it keeps calculating the previously found values of lower Fibonacci sequences. For example, below diagram shows calculating Fib(5) where unnecessarily several same fibonacci values were calculated repeatedly .

For a solution, assume having an array of memory items, where we keep already calculated fib (n) values, then we won't be calculating the same fib(n) values during the intermediate calculation. Thus using this idea, implement a new recursive function that keeps a memory of already calculated fib(n) values.



Solution:

```
memo = {0:0, 1:1}
def fibm(n):
        if not n in memo:
                memo[n] = fibm(n-1) + fibm(n-2)
return memo[n]
```

3. Write an iterative and/or recursive function, which implements the Pascal's triangle:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Solution:

```
def pascal(n):
        if n == 1:
                return [1]
        else:
```

```
            line = [1]
        previous_line = pascal(n-1)
        for i in range(len(previous_line)-1):
            line.append(previous_line[i] + previous_line[i+1])
        line += [1]
        return line
```

4. Write the output of below code segment? Assume the address of z[0][0][0] is 100.

Give a matrix

```
int z[4][2][6] ;
printf(" *(z[1]+1) = %d\n", * (z[1]+1));
printf(" *(z[1][1]+1) = %d\n", * (z[1][1]+1));
```

5.In the class we have seen the upper / lower triangular matrices. We calculated the number of items in lower triangular matrix which was n*(n+1)/2. For a given position u[i][j] we also showed that the address of u[i][j] is at i(i+1)/2 + j.
Do the same calculations for upper triangular matrix and give the number of items and the position of u[i][j].

Solution : number of items do not change

The position is 0 row has n items, 1st row has n-1 items, … i th row has n-i items. We need to add all items including ith row.

n + n -1 + n-2 + … + n-i = n*(n+1)/2 – i*(i+1)/2
= (n^2 + n – i^2 –i)/2

this is number of items before and including the ith row, we need to substract the items after jth position. There are n-1-j more elements on that row. So we need to substract n-1 –j and need to increment 1 for starting 0 index

= (n^2 + n – i^2 –i)/2 – (n-1-j) +1

Sample:

0	1	2	3
	4	5	6
		7	8
			9

i =2 j=2 n=4
16+4-4-2 / 2 – (4-1-2)
14/2 = 7 -1 =6