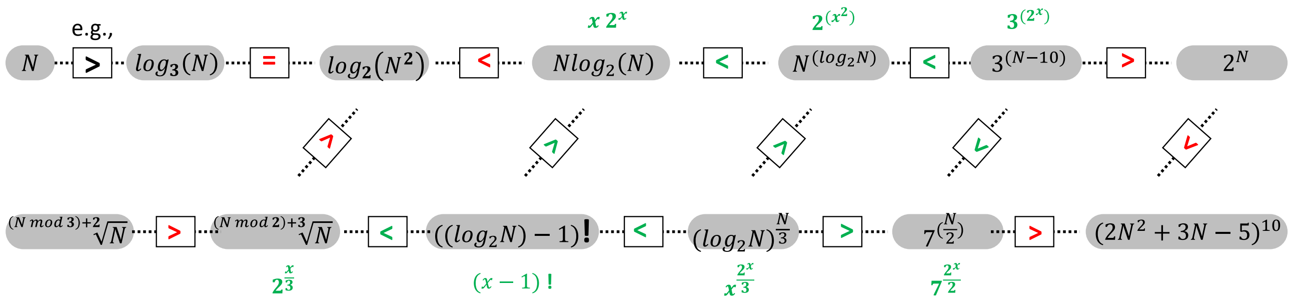
Data Structure Midterm Examination (10410EE 241000)

3:30pm-5:20pm (110 minutes), Nov. 10, 2015

#: ________ Student ID: ___________________ Name: _________________________

- ✧ Please answer questions 1, 2, and 3 (and 10 if appropriate) on the question sheet. For other questions, please answer on the answer sheet in any order.
- ✧ There are 10 questions, each being 11 points.

1. Please compare the asymptotic order of the following time complexity functions (in terms of the worst case) using "=", ">", or "<".



Hints:

- ✧ Substituting *N* with c^x can sometimes ease the comparison.
- ✧ A method to show that $c^N < (N!)$ when *N* is large enough is to observe that $c^N = \mathrm{c}\cdot\mathrm{c}\cdot\ldots\cdot\mathrm{c}\cdot\ldots\cdot\mathrm{c}$ but $(N!) = N\cdot(N-1)\cdot\ldots\cdot\mathrm{c}\cdot\ldots\cdot 1$
  A similar technique may be useful when performing other comparisons.
- ✧ $log_a(a^b) = b, \quad a^{log_a(b)} = b,$

  $log(a\cdot b) = log(a) + log(b), \quad log(a^b) = b\cdot log(a),$

  $log_a(b) = \frac{log_c(b)}{log_c(a)}, \quad (a^b)^c = a^{bc} = (a^c)^b$

2. KMP algorithm

a) Please analyze the failure function for the following patterns.

N	E	E	N	N	E	E	N	E	N	x
0	0	0	1	1	2	3	4	2	1	__1__ if x == 'N' __2__ if x == 'E' __0__ otherwise

b) Please design patterns that exhibit the following failure functions. Please try to compose as long a string as possible and mark an ‘X’ to denote the position (if any) where the failure function becomes invalid.

0	0	1	1	2	0	1	2	3	4	y
a	b	a	a	b	b c	a	b	a	a	__c__ if y == 0 __a__ if y == 1 __x__ if y == 2 __x__ if y == 3 __x__ if y == 4 __b__ if y == 5

3. Please analyze the time complexity of the following algorithm

void func (**int** d1[M][N], **int** d2[N][M])	**Steps per execution**	**Frequency**
{	0	O( )
for (**int** i =0; i<M; i++) {	1	O(M)
Selection_sort (d1[i], N);	N^2	O(M)
}	0	O( )
for (**int** i =0; i<M; i++)	1	O(M)
for (**int** j=0; j<N; j++)	1	O(MN)
d2[j][i] = d1[i][j];	1	O(MN)
for (**int** i =0; i<N; i++) {	1	O(N)
if (d2[0] < d2[M-1])	1	O(N)
Selection_sort (d2[i], M); }	M^2	O(N)
return;	1	O(1)
}	0	O( )
	Overall complexity: O(MN^2+NM^2)	

4. Please prove or disprove

$$F(\mathrm{n}) = \boldsymbol{O}(2^n) \text{ and } G(n) = \boldsymbol{\Theta}(n^2) \quad \Rightarrow \quad log\big(F(n) \times G(n)\big) = \boldsymbol{O}(n \times log(n))$$

$\because F(n) = O(2^n) \quad \therefore \exists c_1, N_1 \in \mathbb{N} \quad \text{s.t.} \quad F(n) \le c_1 2^n \ \forall n \ge N_1$

$\because G(n) = \Theta(n^2) \quad \therefore \exists c_2, N_2 \in \mathbb{N} \quad \text{s.t.} \quad G(n) \le c_2 n^2 \ \forall n \ge N_2$

$\quad = O(n^2)$

$\Rightarrow \log(F(n)\,G(n)) \le \log(c_1 c_2 2^n n^2) \ \forall n \ge N_3$

$N_3 = \max(N_1, N_2)$

$$\begin{aligned}\Rightarrow \log(F(n)\,G(n)) &\le \log(c_1 c_2) + n + 2\log n \\ &\le n\log(n) + n\log(n) + n\log(n) \\ &= 3\,n\log(n) \\ &\quad \forall n \ge \max(N_1, N_2, c_1c_2, 2) \\ &= O(n\log(n)) \quad \#\end{aligned}$$

5. Suppose **D** is a three-dimensional array of one-byte characters. The index of each dimension is a non-negative integer. Suppose **D**[5][4][3] is at address 300 and **D**[6][5][1] at address 182. Please answer the following questions.

a) Is the array in row-major order or column-major order, or both are possible?

b) What are the number of elements in each dimension of **D**? Let us use (x, y, z) to denote that the elements of **D** are arranged as **D**[0… x-1][0… y-1][0… z-1]. If there are many possible answers, please answer like the following:

(x, y, z) = (10, any positive even number, 20) or
(20, any positive odd number, 10)

Please make sure that the previously mentioned **D**[5][4][3] and **D**[6][5][1] are valid indices.

c) What is the address of **D**[1][2][3]? Please give all the possible answers

D[5][4][3] : 300
D[6][5][1] : 182

row major :

$$\alpha_1 + 5yz + 4z + 3 = 300$$
$$-)\ \alpha_1 + 6yz + 5z + 1 = 182$$
$$\Rightarrow \quad -yz - z + 2 = 118 \quad \ast$$

column major :

$$\alpha_2 + 3xy + 4x + 5 = 300$$
$$-)\ \alpha_2 + xy + 5x + 6 = 182$$
$$2xy - x - 1 = 118$$
$$\Rightarrow x(2y-1) = 119 = 1\times119 \text{ or } 7\times17 \text{ or } 17\times7 \text{ or } 119\times1$$

$\Rightarrow x, y = 1, 60 \quad \ast \quad \because$ D[5][4][3]
$7, 9$
$17, 4 \quad \ast \quad \because$ D[5][4][3]
$119, 1 \quad \ast \quad \because$ D[5][4][3]

1) D is column major

2) $(x.y.z) = (7.\ 9.$ any positive int $\geq 4)$

3) D[5][4][3] = 300
D[1][2][3] = K

$$K + (3-3)\cdot 9 \cdot 7 + (4-2)\cdot 7 + (5-1) = 300$$
$$\Rightarrow K = 300 - 18 = \underline{282}$$

6. Please design a program that receives a string with () [] {} and some other characters and checks the **parentheses balance** of the string, i.e., each opening parenthesis has a corresponding closing parenthesis and the pairs of parentheses are properly nested. An example string is as follows.

{[{[[((a+3)*b)] equals [c / 20]}], [data structure is interesting] }

Please use a stack that supports push (adding a character to the stack), pop (removing a character from the stack), and size (reporting the number of elements in the stack) to complete this task.

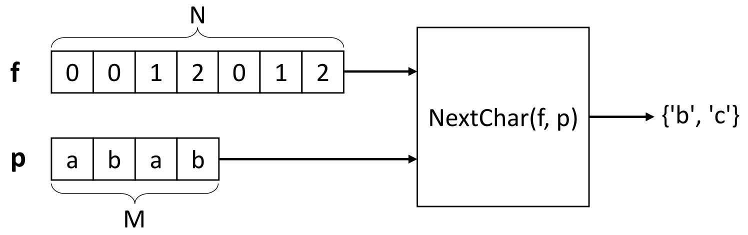
ANS:

```
#include <stack>
using namespace std;

bool ParenthesesBalance(string in)
{
  stack<char> s;
  for (int i=0; i<in.size(); i++){
    switch (char c = in[i]){
      case '(' or '[' or '{': // pseudo code
        s.push(c);
        break;

      case ')' or ']' or '}': // pseudo code
        if (s.size() == 0 || c doesn't match s.pop()) return false; // pseudo code
        break;

      default:
        // do nothing
        break;
    }
  }
  if (s.size()==0) return true;
  else return false;
}
```

7. The KMP algorithm describes how we can derive failure function given a pattern. Reversely, here we want to design an algorithm that can 1) produce a pattern given a specific failure function if such a pattern exist and 2) report an error if such a pattern does not exist.

   a) Please describe your algorithm using pseudo code assuming another algorithm that can drive a character according to the given failure function as follows is available. Hint: consider using recursion to design the algorithm.

```
vector<char> NextChar (vector<int> f, string p);
/* input:
 *   f: An array of N integers
 *   p: A string of M characters whose failure function match
 *   the first M integers of the vector f, 0<M<N.
 * output:
 *   A vector of R candidate character(s), R>=0.
 *   By appending any one of these characters to pattern, the
 *   failure function of the pattern match the first M+1
 *   integers of the vector f.
 * illustration:
 *
 *
 *
 *
 *
 *
 */
```



ANS:

```
void FindPatterns(vector<int> f, string & p)
{
    if (f.size() == p.size()){ // a pattern is found
        cout << p << endl;
        return;
    }

    vector<char> r = NextChar(f, p); // all possible next chars

    for (int i=0; i<r.size(); i++){  // for each possible next char
        p.append(" ");
        p[p.size()-1] = r[i];        // try the char at the end of p
        FindPatterns(f, p);
        p.pop_back();                // undo the append
    }
    return;
}
```

b) Please try to realize NextChar() using pseudo code. In this stage, please **do not** focus too much on the performance of the algorithm.

ANS:

```
vector<char> NextChar(vector<int> f, string p)
{
    vector<char> r;
    int n = p.size();
    if (n == 0) {  // first char
        r.push_back('a');
    } else if (f[n] == 0) {
      char c='a';
      do{
          c++;
          r.push_back(c);
      } while( c has been used in p );  // pseudo code
    } else if (f[n] < n) {
      char c = p[f[n]-1];
      invoke KMP algorithm to check if c is valid // pseudo code
      if (c is valid) r.push_back();
    }
    return r;
}
```

8. Please answer the following questions about object oriented program (OOP)

a) How can OOP help debugging? Please give an example.

- Objects can be individually tested and debugged.
- Re-used objects are typically less prone to bugs, so one can focus on newly implemented objects.
- Member functions are the only interfaces accessing private member data. This narrows down the scope of bugs that related to private member data.
- Object inheritance reduces code redundancy and thus eases debugging.

b) Can OOP help lowering the time complexity of an algorithm? Please give some reasons to support your answer.

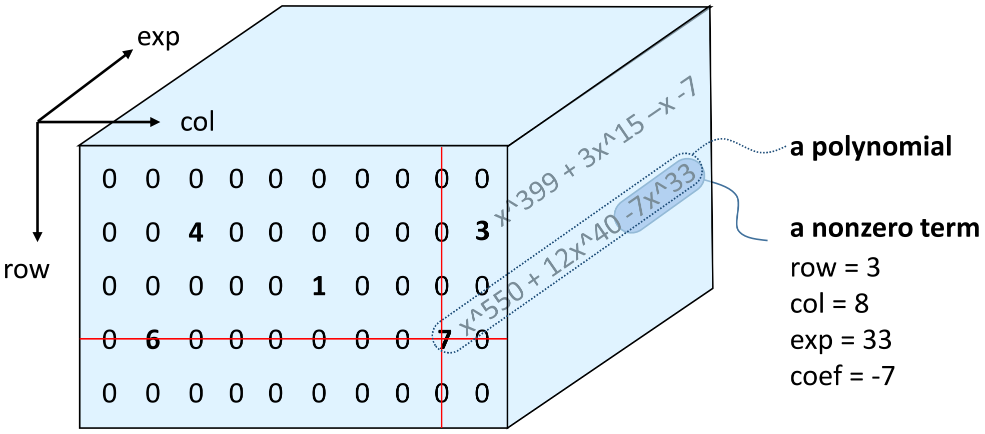
- OOP cannot lower the time complexity of an algorithm because time complexity is an inherited characteristic of an algorithm. For example, Selection Sort has quadratic time complexity in the worst case no matter it is in OOP or non-OOP.
- The other point of view is that any OOP program is eventually compiled into machine code which can be equivalently described using non-OOP languages such as the assembly language. In other words, a non-OOP language always can achieve the same complexity that of an OOP language.

c) Is there any drawback for adopting OOP?

- ✧ Latency of accessing a private data member slightly increases because the need to invoke member functions.
- ✧ Memory usage increases because of the member functions for accessing private data.

9. Please design a **memory efficient** object of **Sparse Matrix of Sparse Polynomials (SMSP)**. By “sparse” we mean a matrix can comprise many zero terms or a polynomial can comprise many zero coefficients. You can use pseudo code to describe your design. Please focus on
1) **constructors**,
2) **destructors**, and
3) a function **adding** two SMSPs.

Idea:
Sparse matrix of sparse polynomials ➔ Sparse 3D matrix



0	0	0	0	0	0	0	0	0	0
0	0	**4**	0	0	0	0	0	0	**3**
0	0	0	0	0	**1**	0	0	0	0
0	**6**	0	0	0	0	0	0	**7**	0
0	0	0	0	0	0	0	0	0	0

a polynomial

a nonzero term
row = 3
col = 8
exp = 33
coef = -7

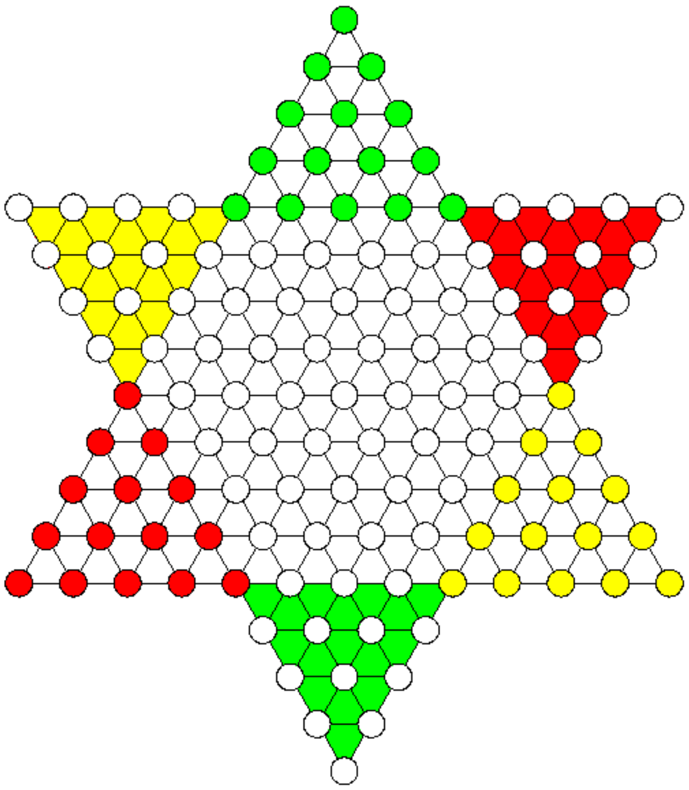
```
class SMSP; // forward declaration
class Term
{
  friend class SMSP;
  int row, col, exp;
  float coef;
};
class SMSP
{
public:
  SMSP();
  ~SMSP();
  SMSP Add(SMSP b);
private:
  int compare(Term a, Term b)
  Term * TermArray;
  int Size;
  int Capacity;
};

SMSP::SMSP()
{
  TermArray = new Term[10];
  Size = 0;
  Capacity = 10;
}

SMSP::~SMSP()
{
  delete [] TermArray;
}
```

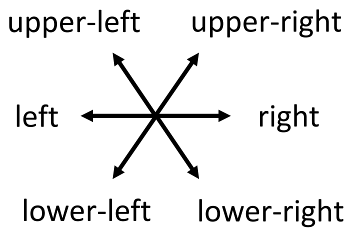
```
int SMSP::compare(Term a, Term b)
{
  if (a.row > b.row)      return 1;
  else if (a.row < b.row) return -1;
  if (a.col > b.col)      return 1;
  else if (a.col < b.col) return -1;
  if (a.exp > b.exp)      return 1;
  else if (a.exp < b.exp) return -1;
  return 0;
}

SMSP SMSP::Add(SMSP b)
{
  SMSP c;
  int ai=0, bi=0;
  while (ai<Size && bi<b.Size) {
    switch (compare(TermArray[ai], b.TermArray[bi]))
    {
      case -1:
        c.NewTerm(TermArray[ai]);
        ai++;
        break;
      case 1:
        c.NewTerm(TermArray[bi]);
        bi++;
        break;
      case 0:
        c.NewTerm(TermArray[ai] + TermArray[bi]); // pseudo code
        ai++; bi++;
    }
  }
}
```

10. Suppose we want to develop a Chinese Checkers program and need **an array representation** of the hexagram-shaped gameboard. Please answer the following questions. In this stage, please **do not** focus too much on the performance and memory efficiency of the algorithm.

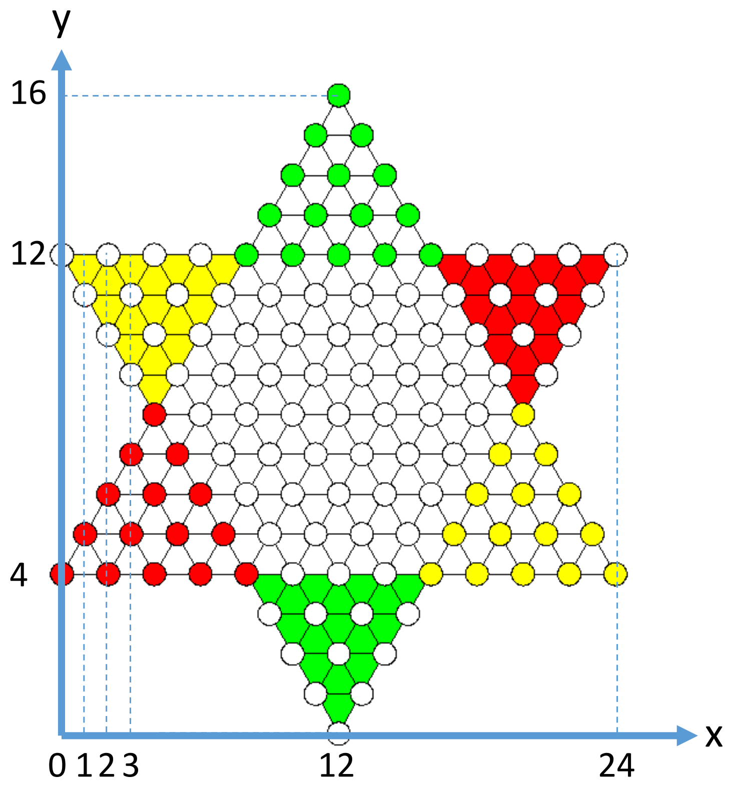
    a) What is your gameboard-array mapping, and what is the required memory space (in bytes) for your gameboard?

    b) How can a checker move? Specifically, how can your program find the array index for a checker taking each of the six moves (upper-left, upper-right, … etc.) and how can your program detect an invalid move exceeding the gameboard boundary?



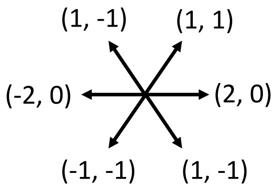


Approach 1:



Checker movement:

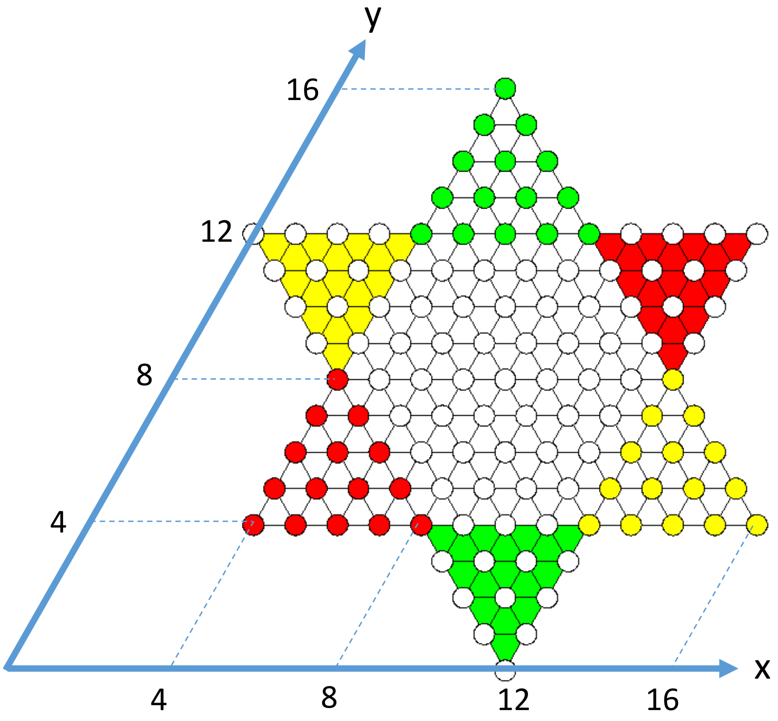
(Δx, Δy):



Boundary check:
(y<4 && x+y<12)
(y<4 && x-y>12)
....

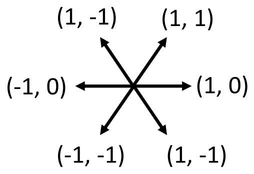
Memory usage:
17 * 25 * 2 bits

Approach 2:



Checker movement:

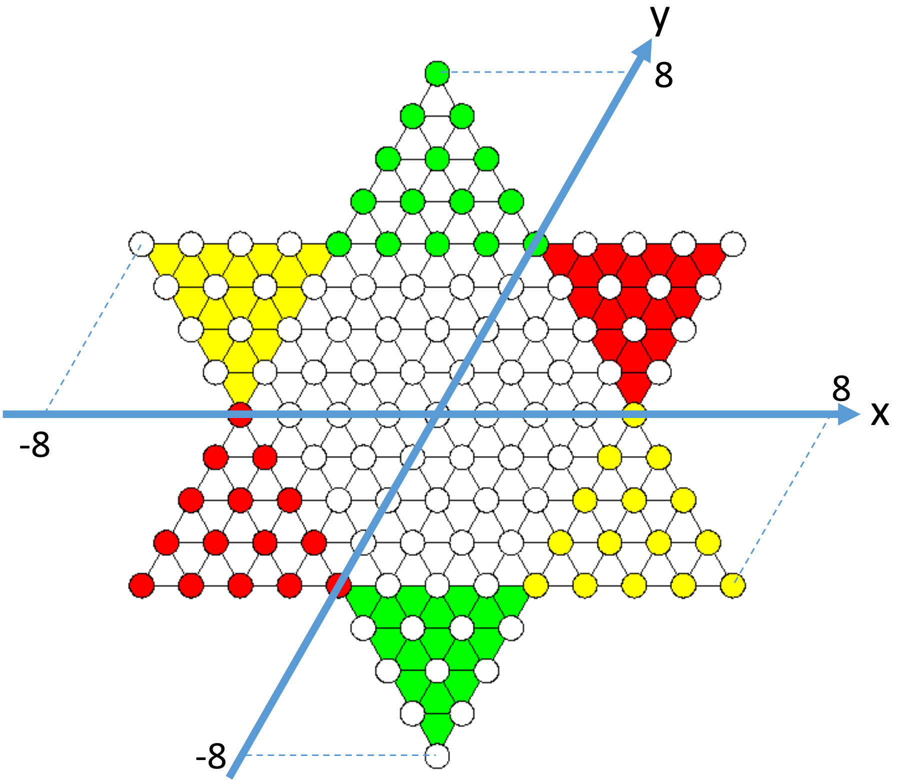
(Δx, Δy):



Boundary check:
omit

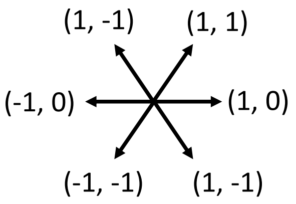
Memory usage:
17 * 17 * 2 bits

Approach 3:



Checker movement:

(Δx, Δy):



Boundary check:
omit

Memory usage:
17 * 17 * 2 bits