Department of Computer Science and Engineering
National Sun Yat-sen University
Data Structures - Middle Exam, Nov. 16, 2015 參考解答

1. Suppose that the first element of array *a* is *a*[0][0] or *a*[0][0][0] and its address is 200. Assume that each *int* element requires 4 bytes and each *float* element requires 8 bytes. Please give the address of the indicated element in each of the following cases. (12%)
   (a) *int a*[7][10]; row-major order; find element *a*[4][5].
   (b) *float a*[7][10]; column-major order; find element *a*[4][5].
   (c) *int a*[5][4][6]; column-major order; find element *a*[3][1][4].
   (d) *float a*[5][4][6]; row-major order; find element *a*[3][1][4].

   Ans:
   (a) 200+(4*10+5)*4 = 380
   (b) 200+(5*7+4)*8 = 512
   (c) 200+(4*5*4+1*5+3)*4 = 552
   (d) 200+(3*4*6+1*6+4)*8 = 856

2. Suppose that we have a declaration *char c*[ ]=”BED”. How many bytes are allocated? What is the real content in the memory? (6%)
   Ans:
   4 bytes;
   c[0] = $(01000010)_2 = (42)_{16} = 66$
   c[1] = $(01000101)_2 = (45)_{16} = 69$
   c[2] = $(01000100)_2 = (44)_{16} = 68$
   c[3] = $(00000000)_2 = (00)_{16} = 0$

3. Suppose that A, B, C are matrices of size $n\times n$. What is the time complexity for matrix multiplication $C=A\times B$? Please derive your answer. (6%)
   Ans:
   $O(n^3)$;
   以 C(i, j)表示 C 矩陣第 i 列，第 j 行的元素；則 C(i, j) = A(i, 1)*B(1, j) + A(i, 2)*B(2, j) + A(i, 3)*B(3, j) + … + A(i, n)*B(n, j)。要得到 C 矩陣裡的一個元素要計算 n 次乘法和 n-1 次加法，所以得到 C 矩陣中一個元素的時間複雜度是 O(n+n-1) = O(n)。C 矩陣中共有 n*n 個元素，所以矩陣相乘總體時間複雜度是 $O(n^3)$。

4. What are printed by each of the following C programs? (8%)
   (a) char d=14;   printf("%d \n",((d << 2) >> 3));
   Ans: 7
   d = (00001110)2 → (d<<2) = (00111000)2 → ((d<<2)>>3) = (00000111)2 = 7.
   (b) int c[ ]={20,21,22,23,24};   int *r;
   r=c+1;   *(++r)=c[0]+6;

```
        printf("%d %d %d %d \n",c[1],c[2],*r,*(r+1));
```

Ans: 21 26 26 23

r=c+1;等同於r=&c[1];代表讓指標r指向c[1]的位置。

*(++r)=c[0]+6;可以拆解成兩行來看：先執行r=r+1;再算*r=c[0]+6;。r=r+1;等同於r=c+2;等同於r=&c[2]; 代表讓指標r改指向c[2]的位置。接著再執行*r=c[0]+6;此時r指向c[2]的位置，把c[2]的值設為c[0]+6。最後，r的值等於c+2，代表r指向c[2]的位置，r+1就指向c[3]的位置。

5. The Hanoi towers problem: There are 3 towers and n , $n \geq 1$, disks of different diameters placed on the first tower. The disks are in order of decreasing as one scan up the tower. Monks were supposed to move the disks from tower 1 to tower 3 obeying the following rules: (a) only one disk can be moved at any time and (b) no disk can be placed on top of a disk with smaller diameter. Please derive the number of disk moves required for accomplishing this task. (12%)

Ans:

把 n 個 disks 從 tower 1 移動到 tower 3 的過程可以可分解成三個步驟：(1)把 tower 1 頂端 n-1 個 disks 搬到 tower 2，(2)把 tower 1 僅剩的一塊 disk 搬到 tower 3，(3)把 tower 2 全部 n-1 個 disks 搬到 tower 3。步驟 2 只搬動一塊 disk，所以所需的搬動次數僅 1 次。步驟 1 或 3 則需要搬動 n-1 塊，搬動 n-1 塊 disks 的移動次數可以遞迴地用上述方法來計算。

$T(1) = 1$, $T(n) = 2*T(n-1)+1 = 2*(2*T(n-2)+1)+1 = 2^2*T(n-2)+2+1 = 2^2*(2*T(n-3)+1)+2+1 = 2^3*T(n-3)+2^2+2+1 = 2^{n-1}*T(1)+2^{n-2}+\ldots+1 = 2^n-1$

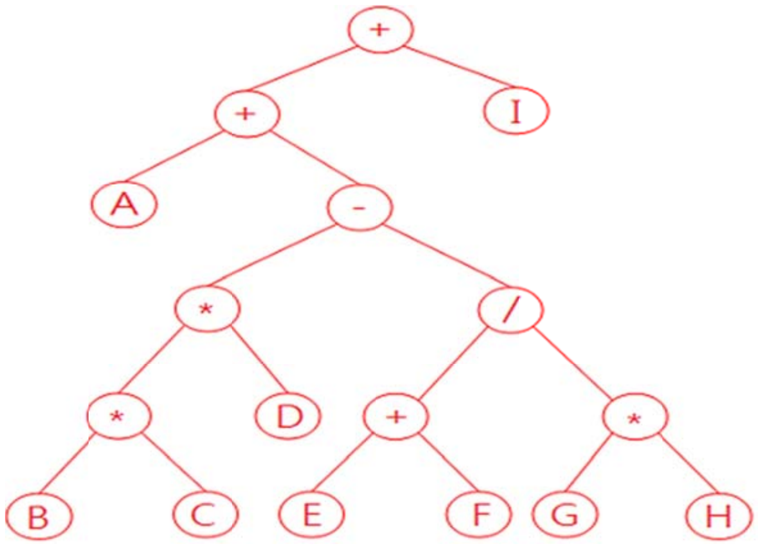
搬動 n 塊 disks 所需移動的次數是 2^n-1 次。

6. Transform the *prefix* expression ++*A*−***BCD*/+*EF***GHI* to *infix* and *postfix* expressions. Draw its expression tree. (10%)

Ans:

Infix: A+[B*C*D-(E+F)/(G*H)]+I

Postfix: ABC*D*EF+GH*/-+I+



7. Please present the method, with the help of a stack, to evaluate the value of a postfix expression. Use 6 2 3 + - 3 8 2 / + * 2 * 3 - as an example to explain your

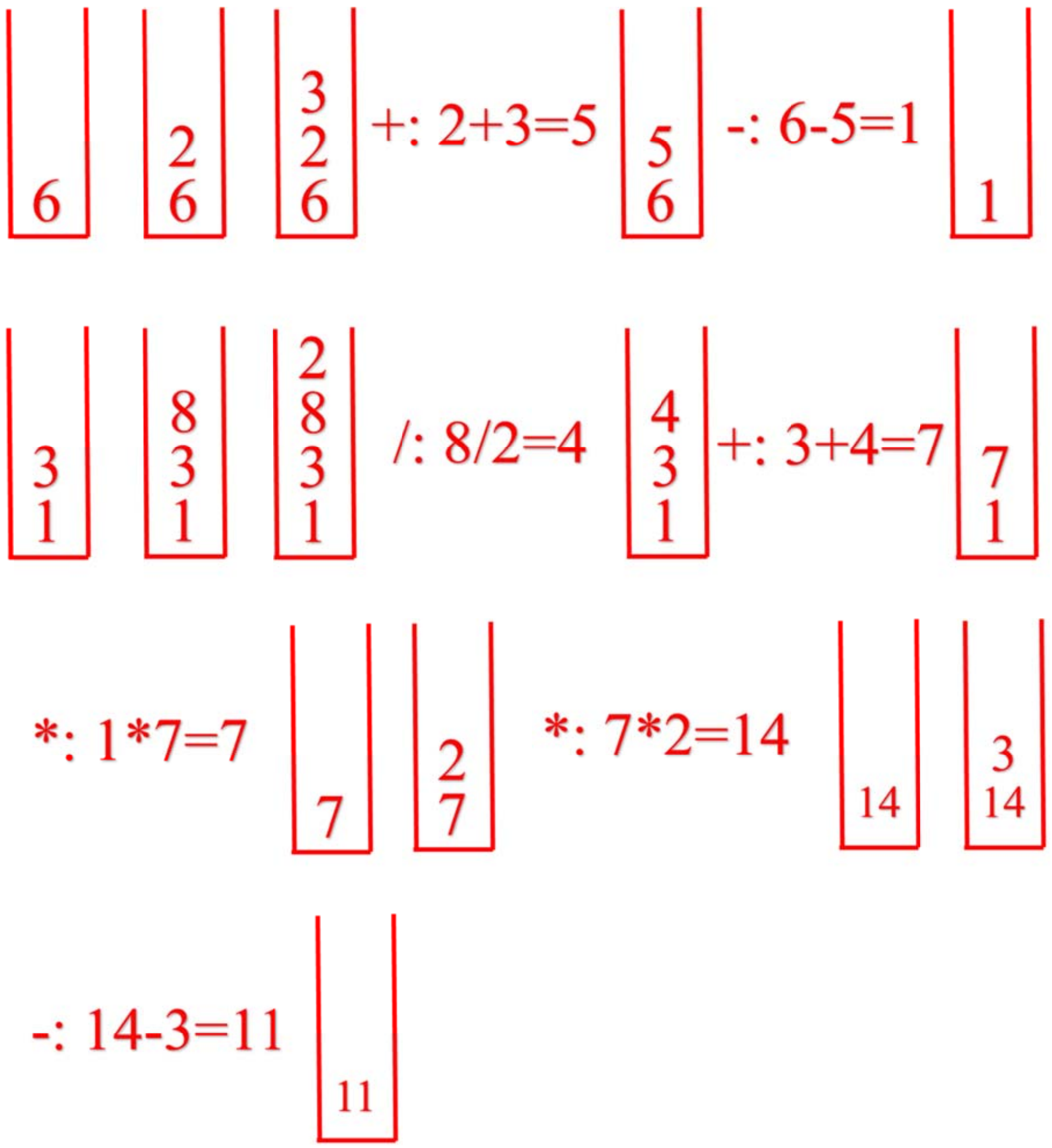
method, where each digit (“2”, “3”, “6”, “8”) represents an operand. Note that you have to describe your method. If there is no description of your method, you will get no point. (10%)

Ans:

使用 stack 計算 postfix expression 的規則：由左至右讀入 token

(1) 遇數字(operand)，存入 stack

(2) 遇 operator，將 stack 最上層兩個數字進行運算，再將計算結果放入 stack。



8. Write a recursive C/C++ function to perform the binary search on an increasingly sorted array. (12%)

```
int BSearch(int a[ ], int x, int left, int right)
// search for x
```

```
//a: increasingly sorted array,      left, right: left and right indexes for search
//Return the index if found. Return -1 if not found.
{
```

Please write the body of BSearch( ).

```
} // end of BSearch( )
```

Ans:

```
int BSearch(int a[ ], int x, int left, int right)
{
      if (left <= right) {
       int middle = (left + right)/2;
       if (x < a[middle])
          return BSearch(a, x, left, middle-1);
       else if (x > a[middle])
          return BSearch(a, x, middle+1, right);
       else return middle;
    } // end of if
    return -1;// not found
} // end of BSearch( )
```

9. Write a C/C++ function to perform *IsEmpty* and *Push* operations of a circular queue implemented with an array. (12%)

```
int front, rear;            // front, rear pointers
int capacity=100;           // size of queue
char q[100];                // array for the circular queue
bool IsEmpty( )
   // Return true if q is empty.
{
```

(a) Please write the body of IsEmpty( ).

```
} // end of IsEmpty( )
void Push(char x)
// Insert x into q at the rear. You have to check if q is full before the insertion.
{
```

(b) Please write the body of Push( ).

```
} // end of Push( )
```

Ans:

```
bool IsEmpty()
{
      return front == rear;
} // end of IsEmpty( )
```

```
void Push (char x)
// Insert x into q at the rear. You have to check if q is full before the insertion.
{
      if ((rear + 1) % capacity == front)
            throw "Queue is full."
       rear = (rear + 1) % capacity;
       queue[rear] = x;
} // end of Push( )
```

10. Write a C++ function to reverse a singly linked list. For example, suppose that the give list X=(x_1, x_2, …, $x_{n\text{-}1}$, x_n). After the reversing process, the list will become (x_n, $x_{n\text{-}1}$, …, x_2, x_1). (12%)

```
class ChainNode {
   int data;
   ChainNode *link;
};
class Chain {
   ChainNode *first;      // first node of the list
   void reverse( )
        // Reverse the list.
   {
   ChainNode *p, *c;          //    p:previous, c:current
```

Please write the body of reverse ( ).

```
   } // end of reverse ( )
};
```

Ans:

```
void reverse( )    // Reverse the list.
{
   ChainNode *p, *c;          //    p:previous, c:current
   c = first
   p = 0;      // before current
   while (c) {
        ChainNode *r = p;
          p = c;
          c = c ->link;     // moves to next node
          p->link = r; // reverse the link
   }
   first = p;
} // end of reverse ( )
```