Department of Computer Science and Engineering
National Sun Yat-sen University
Data Structures - Middle Exam, Oct. 28, 2024

1. What are printed by each of the following C programs? (20%)

   (a)
   ```
   int a=24, b=14;
   printf("%d    %d\n", a&(-a), b&(-b)); // &: bitwise AND, ~: bitwise NOT
   ```

   (b) What is the purpose of the program in (a)?

   (c)
   ```
   int c = -1;
   printf("%d    %d    \n", (c << 4) + 4, c >> 24);
   ```

   (d)
   ```
   int g(int n) {
      if (n <=1)    return n;
      else if (n%2==0)    return g(n-1)+g(n/2);
      else return g(n-1)+1; }
   void main( ) { printf("%d %d    \n", g(5), g(6));}
   ```

   (e)
   ```
   void f(int w[ ], int x[ ], int *y, int *z) {
      *y=15;    *(++y)=25;
      printf("%d    %d    %d    %d \n", w[1], x[0], *(y+4), *z);    }
   int main( ) {
      int a[ ]={40,41,42,43,44,45,46,47,48,49,50};
      f(a, a+4, &a[0]+3, &a[5]); }
   ```

2. Let *p*, *q* and *r* be three Boolean variables. When one variable is "True", it is represented by 1; when one variable is "False", it is represented by 0. Please list the truth table of (IF *p*, THEN *q*) AND (IF *q*, THEN *r*). (12%)

3. The length of a horizontal wire is 100 cm (the leftmost scale is 0 and the rightmost is 100). There are *n* ants at different starting positions (represented as numbers between 0 and 100). Each ant crawls to the left or right with the same speed 1 cm/sec. Some ants crawl to the left and others crawl to the right. When two ants meet on the way, each crawls back in the opposite direction. When an ant crawls to the left or right end, it will fall down. You are asked to answer the time when the last ant falls down, where the starting time is set to 0 second. For example, *n*=2, the starting positions are (30, right), (60, left), which means that an ant crawls to the right at 30 cm, and an ant crawls to the left at 60 cm. The answer is 70 seconds. Now, *n*=5, the starting positions are (25, left), (30, right), (40, right), (50, left), (75, left). What is the answer? (8%)

4. An array is declared as *a*[5][6]. Now we have an element *a*[*x*][*y*]. Suppose row-major addressing and column-major addressing are used. What are the values of *x* and *y* so that *a*[*x*][*y*] locates at the same address when the two addressing methods are used? (10%)

5. Please present a method for checking whether a postfix is valid. (10%)

6. Explain each of the following terms. (16%)
   (a) the concept of function call by using a stack
   (b) Hanoi tower problem
   (c) reference count in a generalized list
   (d) inheritance in C++ language

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

```
… Binary(….)
// Binary search on a sorted array. Please define the parameters for the function
and the returned value
{
```

Please write the body of Binary ( ).

```
} // end of Binary ( )
```

8. Write a C++ function to insert an integer into a singly linked list, which is sorted with a nondecreasing order. For example, suppose that the given list $X=(x_1, x_2, \ldots, x_{n-1}, x_n) = (3, 6, 7, 7, 9)$. After the insertion of an integer 8, the list will become (3, 6, 7, 7, 8, 9). You have to involve the following possible cases: the list may be empty, the inserted element may be inserted as the first or last node. (12%)

```
class ChainNode {
   public:
   int data;
   ChainNode *link;
   public:   ChainNode (int element = 0, ChainNode *next =0)   // constructor
   {   data = element;        link = next;   }
};   // end of class ChainNode
class Chain {
   ChainNode *first;      // first node of the list
   void insert(int a)   // Insert an integer a into the list.
   {
```

Please write the body of insert( ).

```
   } // end of insert( )
};   // end of class Chain
```

Answers:

1. (a) 8 2
   (b) 輸出 a 和 b 在二進位表示法，最低的非零位元的值
   (c) -12 -1
   (d) 6 9
   (e) 41 25 48 45

Explanation:

(a) 24 = $(11000)_2$
   -24 = 2's complement = $(111\ldots11101000)_2$
   $(11000)_2$ & $(111\ldots11101000)_2$ = $(01000)_2$ = 8

   14 = $(01110)_2$
   -14 = 2's complement = $(111\ldots11110010)_2$
   $(01110)_2$ &$(111\ldots11110010)_2$ = $(00010)_2$ = 2

(b) 輸出 a 和 b 在二進位表示法，最低的非零位元的值。
   a = 24, b = 14;
   program a 輸出的 8 和 2 分別代表了 24 和 14 這兩個數字在其二進位表示中最右側的非零位元的值，因為 24 和 14，僅有最右側的非零位元瑜運算後，得到 1。
   24 = $(11000)_2$
   8 = $(01000)_2$
   14 = $(1110)_2$
   2 = $(0010)_2$

(c) -1 = $(111\ldots111)_2$
   Left-shift 4 = $(111\ldots1110000)_2$ = -16，向左移位後，右側補進來的是 0。
   (-16) + 4 = -12

   -1 = $(111\ldots111)_2$
   Right-shift 24 = $(111\ldots1111111)_2$ = -1，向右移位後，右側補進來的是最左之位元。

(d)
   g(1)=1
   g(2)=g(1)+g(1)=2
   g(3)=g(2)+1=3
   g(4)=g(3)+g(2)=5
   g(5)=g(4)+1=6
   g(6)=g(5) +g(3)=9

(e)

f(a, a+4, &a[0]+3, &a[5])
w: 指向 a[0]的位址
x: 指向 a[4]的位址
y: 指向 a[3]的位址
z: 指向 a[5]的位址

f 內部運行:
*y = 15; // y 一開始指向 a[3]，所以 a[3] = 15
*(++y); // y 後移一格，現在指向 a[4]，所以 a[4] = 25
修改後的 array a[] = {40, 41, 42, 15, 25, 45, 46, 47, 48, 49, 50}

printf("%d %d %d %d \n", w[1], x[0], *(y+4), *z);
// w 指向 a[0]的位址，w[1] 等同於 *(w+1)，對應到 a[1] = 41
// x 指向 a[4]的位址，x[0]等同於*(x+0)，對應到 a[4] = 25
// *(y + 4)，y 目前指向 a[4]，所以 y+4 對應到 a[8] = 48
// *z 指向 a[5] = 45
// 最終輸出: 41 25 48 45

2.

p	*q*	*r*	IF *p*, THEN *q*	IF *q*, THEN *r*	(IF *p*, THEN *q*) AND (IF *q*, THEN *r*)
			(NOT *p*) OR *q*	(NOT *q*) OR *r*	((NOT *p*) OR *q*) AND ((NOT *q*) OR *r*)
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1

3. 75 秒

模擬：將 5 隻螞蟻進行編號為 1 至 5，然後模擬行進時，若碰到另一隻螞蟻，則各自回頭。此種模擬可以計算出最後一隻螞蟻掉落的時間，但是模擬過程過於複雜。

簡化模擬：模擬行進時，若碰到另一隻螞蟻，各自回頭，可以想像成兩隻螞蟻各自穿透而過，也就是兩隻螞蟻各自前進，而不是各自回頭。可以簡化成穿透而過的方式，主要是原先螞蟻遇到另一隻螞蟻會各自回頭，而不問其編號。

簡化模擬後，走最遠的螞蟻就是最後掉落的螞蟻，也就是(75, left)，其答案為 75 秒。

4.
Answer: x=4, y=5.
Explanation:

Row-major addressing: 6x+y
Column-major addressing: 5y+x
We have 6x+y=5y+x
Then 5x=4y. Thus x=4 and y=5.

5.
方法一：
建立一個空 stack，從左到右遍歷式中的每個元素。
若為運算資料，push it onto the stack，
若為運算符號，檢查 stack 中是否至少有兩個元素，若有則 pop 出兩個元素進行運算，再將結果推回 stack；若不足兩個元素，則為 invalid。
遍歷結束後，stack 若剩一個元素，則為 valid；否則為 invalid。

方法二：
設置一個 counter，初值為 0，從左到右遍歷式中的每個元素。
若為運算資料，則將 counter 加 1；
若為運算符號，則將 counter 減 1。
計算過程過程，counter 不小於 1， 且最後為 1，則為 valid；否則為 invalid。

6.

(a) Dynamic allocation on the stack is used for local variables and parameters when a function is called. Once the function completes, this allocated storage is automatically reclaimed by the system.

(b) There are n disks, each with a distinct diameter. Larger disks must always be placed below smaller ones. The objective is to move all disks from peg A to peg C, using peg B as an auxiliary. Only one disk can be moved at a time.

(c) The reference count of an object stores the number of pointers (references) pointing to the object. When the reference count becomes 0, the object is no longer in use and can be safely removed.

(d) inheritance in C++ language: a feature that enables new classes to be created from existing ones, allowing the new classes to inherit properties and behaviors from the parent classes.

7.

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

8.

Method 1:

```
void insert(int a)
{
     ChainNode* newNode = new ChainNode(a,NULL);    // create a new node

     if(first == NULL)    /// insert into an empty list
         first = newNode;
     else if (first->data >= a)      // inserted as the first node
         newNode->link = first;
         first = newNode;
     else {
         ChainNode* p = first->link;
         ChainNode* q =first;
         while (p != NULL && p ->data < a) {    // find correct position, insert after q
              q = p;
              p = p->link;        // move to next node
         }
         newNode->link = p;      // insert new node
         q->link = newNode
      }      // end of else
      Return;
}     // end of insert( )
```

Method 2:

```
void insert (int a)
{
     ChainNode* newNode = new ChainNode(a);    // create a new node

     if (first == NULL) {  // insert into an empty list
```

```
        first = newNode;
    }
    else if (a < first->data) {  // inserted as the first node
        newNode->link = first;
        first = newNode;
    }
    else {
        ChainNode* c = first;
        while (c->link && c->link->data <= a) {   // find the correct position
            c = c->link;
        }
        newNode->link = c->link;
        c->link = newNode;
    }
} // end of insert( )
```