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

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].

2. Suppose that we have a declaration *char c*[ ]=”BED”. How many bytes are allocated? What is the real content in the memory? (6%)

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%)

4. What are printed by each of the following C programs? (8%)
   (a)
```
char d=14;    printf("%d \n",((d << 2) >> 3) );
```
   (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));
```

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%)

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

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 (“6”, “2”, “3”) 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%)

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( )
```

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( )
```

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