COSC 1437 (DL)- Fall 2016
Quiz 4- Chapters 16,17, 19 & Notes
Total Points: 30

Due: Tuesday December 13th @ 11:59PM. Look at Syllabus/ICR about late work.

Directions: For Questions 1-15, clearly mark answers on a separate word (or notepad) document. See sample file/directions provided by your professor and submit to the appropriate location on the MyTCC (BlackBoard) site.

– Assume all variables are properly declared- unless otherwise mentioned.

Multiple Choice. Mark the one best answer for each question. **(2 pts. each)**

1. Complete the following function template:

```
____
T reverse(T x)
{
     return -x;
}
```

A. `template : class T`
B. `template <class T>`
C. `template (class T)`
D. `template {class T}`

2. Consider the accompanying definition of a recursive function:

```
void printNum(int num)                        //Line 1
{                                             //Line 2
    if (n < 0)                                //Line 3
        cout << "Num is negative" << endl;    //Line 4
    else if (num == 0)                        //Line 5
        cout << "Num is zero" << endl;        //Line 6
    else                                      //Line 7
    {                                         //Line 8
        cout << num << " ";                   //Line 9
        printNum(num - 1);                    //Line 10
    }                                         //Line 11
}                                             //Line 12
```

Which of the statements represent the base case?

A. Statements in Lines 3 and 4
B. Statements in Lines 5 and 6
C. Statements in Lines 3-6
D. Statements in Lines 5-10

3. Consider the accompanying definition of a recursive function:

```
int puzzle(int start, int end)
{
    if (start > end)
        return start - end;
    else if (start == end)
        return start + end;
    else
        return end * puzzle(start + 1, end - 1);
}
```

What is the output of the following statement?

```
cout << puzzle(3, 7) << endl;
```

A. `10`
B. `21`
C. `42`
D. `420`

4. The `try` block is followed by one or more ____ blocks.
A. `catch`
B. `finally`
C. `do`
D. `throw`

5. If you try to solve a problem recursively, you should
A. find a recursive call that will lead towards one of the stopping cases.
B. find all the stopping cases and the values if needed at that case.
C. All of the above.
D. None of the above.

6. Writing a template class
A. means we never have to write non-template classes again.
B. allows us to write one class definition that can hold different data types.
C. allows us to skip the implementation of that template class.
D. is illegal.

7. Two types of container classes in the STL are
A. box and cylinder
B. array and struct
C. sequential and associative
D. constant and literal

8. To catch an exception, a program must
A. first experience an unrecoverable error.
B. have a `throw()` function.
C. have a `try/catch` construct.
D. first experience a fatal error.

9. If the following function will throw a `string` exception, then

```
void myFunction();
```

A. the function definition and declaration should have a throw list.
B. the function should have an empty throw list.
C. the function definition, but not the declaration should have a throw list.
D. All of the above.

10. A `catch` block ____ display, or to use in any way, what is thrown.
    A. is required to
    B. is not required to
    C. must not
    D. cannot

11. A linked list is a collection of components called ____.
    A. elements
    B. nodes
    C. members
    D. pointers

12. What is the effect of the following statements?

```
struct nodeType
{
    int info;
    nodeType *link;
};

nodeType *head, *p, *q, *newNode;
newNode = new nodeType;

newNode->info = 50;
```

    A. Stores 50 in the info field of the `newNode`
    B. Creates a new node
    C. Places the node at location 50
    D. Cannot be determined from this code

13. What is special about the last node in a dynamic linked list?
    A. Its component (data) member is empty.
    B. Its component (data) member contains the value 0.
    C. Its link member is empty.
    D. Its link member contains the value `NULL`.

14. Given the template:

```
template<class T>
T Half(T n)
{
  return n / 2;
}
```

Which of the following does not contain a valid function call?
    A. `anInt = Half<int>(324);`
    B. `aFloat = Half<float>(98.6);`
    C. `anInt = Half(324);`
    D. `anInt = Half<324>;`

15. You typically do not throw an exception
    A. when invalid data is detected
    B. to test an exception handler
    C. when the processing that's done by a method can't be completed
    D. after catching an exception and performing some processing

Extra Credit: Implement the following program. Follows same program guidelines and graded on the same scale as program sets. Submit only your `.cpp` file- no test runs/folder required. Partial credit given. (**10 points)**

The game of "Jump It" consists of a board with n positive integers in a row except for the first column, which always contains zero. These numbers represent the cost to enter each column. Here is a sample game board where n is 6:

0	**3**	**80**	**6**	**57**	**10**

The goal of game is to move from the first column to the last column in the lowest total cost. The number in each column represents the cost to enter that column. The game always begins in the first column and have two types of moves. You can either move to the adjacent column or jump over the adjacent column to land two columns over. The cost of a game is the sum of the costs of the visited columns. In the board shown above, there are several ways to get to the end. Starting in the first column, our cost so far is 0. One could jump to 80, then jump to 57, then move to 10 for a total cost of `80 + 57 + 10 = 147`. However, a cheaper path would be to move to 3, jump to 6, then jump to 10, for a total cost of `3 + 6 + 10 = 19`.

Write a C++ recursive solution to this problem. The user will enter the size of the board and its values. Output the actual sequence of jumps and cheapest (optimal) cost of this sequence. After making sure that your solution works on small arrays, test your solution on boards of larger and larger values of n to get a feel for how efficient and scalable your solution is. A user defined function must be used in the program. Output should be user friendly.

Name the program: `JumpItXX.cpp`, where XX are your initials.