# USC CSCI 103L Fall 2025 Final Exam Sol

Saturday December 13th 11am

Name: _____

USC ID: _____

USC email: _____

**Please also write your name and ID on the top of each page!**

Please Circle the section in which you are registered:

Prof. Goodney 9:30 AM

Prof. Goodney 11:00 AM

Exam Notes:

Running time: 110 minutes

Allowed Resources: 1 8.5” x 11” hand-written, double sided “cheat sheet”

**NO OTHER ALLOWED RESOURCES**

Total Points possible: 100

Please read the entire exam before beginning.

USC ID_____ Name_____

## Question 1 (25 points) True/False (Circle the correct answer)

1. **T** / **F** For a derived class using the constructor initialization list can be used to explicitly call the base class constructor.

2. **T** / **F** If no copy constructor is provided, the compiler will automatically generate a default copy constructor.

3. **T** / **F** Operator overloading in C++ allows existing operators to be implemented or redefined to work with user-defined data types.

4. **T** / **F** C++ does not differentiate between a pointer that points to a single variable and a pointer that points to a 1-d array.

5. **T** / **F** When you allocate a multidimensional array the computer sets aside a total amount of memory, but it is up to you how to interpret the dimension ordering.

6. **T** / **F** In C++, the default access for both structs and classes is public.

7. **T** / **F** Constructors in C++ are invoked explicitly by the programmer whenever an object of a class is created.

8. **T** / **F** Objects fuse data and functions that operate on that data into one entity.

9. **T** / **F** With C-strings, operators like comparison (== or !=), assignment/copy (=), and append/concatenate (+) are usable (i.e they will compare, copy or concatenate the strings).

10. **T** / **F** When calling the functions from #include <cstring> that modify a C-string, the library functions will always add the NULL character termination at the end of the string.

11. **T** / **F** Integers and doubles passed as command line arguments can't be used directly (they need to be converted from strings to ints/doubles).

12. **T** / **F** We should check for the self assignment case in the copy constructor.

13. **T** / **F** Copy semantics in C++ involve how objects are duplicated or cloned when passed by value or assigned, and it includes the concepts of shallow and deep copies.

14. **T** / **F** If a class does not have any pointers as data members, you will likely not need to define a copy constructor.

USC ID_____ Name_____

15. **T** / **F** When an object of a user-defined type is passed by value to a function, the copy constructor is invoked.

16. **T** / **F** The following command is sufficient for dynamically allocating a 2-D array:
int **array = new int*[n];

17. **T** / **F** When throwing an exception you **must** throw an object that is part of the std::exception hierarchy.

18. **T** / **F** You should use the throw statement when some code has encountered a problem but cannot handle that problem itself.

19. **T** / **F** Exceptions in C++ provide a way to handle abnormal or unexpected situations that arise during program execution, allowing for control to be transferred to a suitable catch block for handling.

20. **T** / **F** When a try block has more than one catch block, the order in which the catch blocks appear does not affect the way the code functions.

21. **T** / **F** In C++, when using exception handling, it is mandatory to catch all exceptions that may be thrown within a try block.

22. **T** / **F** The ifstream type defaults to allowing both read and write access to a file.

23. **T** / **F** The following code snippet will create a new file if it does not exist.
ofstream myfile("f.txt");

24. **T** / **F** When using the seekg() function, the current position of the file stream can be set.

25. **T** / **F** If a base class (type Base) has a virtual function f(), and a derived class (type Child) overrides that function, it is possible to achieve runtime polymorphism with an object instance of type Base.

USC ID_____ Name_____

## Question 2 (15 points) Multiple Choice (tick/mark the correct answer)

Each question has only ONE correct answer. If you change your answer, mark that change as CLEARLY as possible!

1. Which of the following options describes the correct usage of getline()?
   - ☐ For reading a single character
   - ☑ ~~For reading a whole line of input, including spaces~~
   - ☐ For reading a word from a file
   - ☐ For clearing an input stream

2. Which of the following statements about std::vector and std::deque is correct:
   - ☑ ~~std::vector is efficient at inserting elements at both ends~~
   - ☐ std::deque supports efficient insertion at both ends
   - ☐ std::deque is more memory efficient than std::vector
   - ☐ std::vector supports bidirectional iterators, but std::deque does not

3. Which of the following options describes a facet of polymorphism?
   - ☐ A function can have multiple overloaded versions
   - ☑ ~~A pointer of a base class can point to an object of a derived class~~
   - ☐ A function can call itself recursively
   - ☐ It is not allowed to define variables with the same name in the same scope

4. In C++, ifstream is used to:
   - ☐ Write to a file
   - ☑ ~~Read from a file~~
   - ☐ Process standard input
   - ☐ None of the above is correct

5. Which of the following options is correct about operator overloading?
   - ☐ operator<< can only be implemented as a global function
   - ☐ operator+ must be implemented as a member function
   - ☑ ~~Operator overloading cannot change the priority of the operands~~
   - ☐ Operator overloading cannot be used with friend functions of a class

USC ID_____ Name_____

6. Which expression calls the default constructor of Object:
   - ☐ Object * a = new Object();
   - ☐ vector<Object> v(10);
   - ☐ Object a;
   - ☑ ~~All of the above~~

7. What property of arrays allows pointer arithmetic to "walk" through an array?
   - ☑ ~~contiguous allocation~~
   - ☐ static size allocation
   - ☐ unknown number of elements
   - ☐ access by index
   - ☐ arrays are passed by pointer

8. Arrays can implement a ___, as in a mapping between positive integers and values in some other domain.
   - ☑ ~~function~~
   - ☐ allocation
   - ☐ compilation
   - ☐ declaration

9. Which requirement do the following cstring library functions share?
   char *strcpy(char *dest, const char *src)
   char *strcat(char *dest, const char *src)
   - ☐ They both require the destination string to be longer than the source string.
   - ☑ ~~They both expect that the src c-strings are NULL-terminated.~~
   - ☐ They both require the source string to have a larger size than the destination string.
   - ☐ They both won't compile unless both the source and destination strings are NULL-terminated.
   - ☐ They both require the source string to be a substring of the destination string.

USC ID_____ Name_____

10. When is the operator=() invoked, and what is its primary purpose?
    - ☐ It is called during the declaration of an object, such as MyArray a2 = a1, and it is responsible for creating a new object.
    - ☑ ~~It is invoked when an object already exists and the object is the LHS of the '=' operator. Its purpose is to define how the assignment (copy) should be made.~~
    - ☐ It is exclusively utilized in the copy constructor when creating a new object.
    - ☐ It is called when you declare a new object without assigning any value to it, ensuring proper initialization.

11. In C++, if a class has a user-defined copy constructor, when is it invoked?
    - ☐ When the class is instantiated without initialization
    - ☑ ~~When a class instance is passed by value to a function~~
    - ☐ When a class instance is passed by reference to a function
    - ☐ When a class instance goes out of scope

12. What would happen with the following code given that the classes are implemented correctly with all the necessary components:

```
Student s1("Chad", 90);
Student s2 = s1;
```

    - ☐ Call the assignment operator
    - ☐ Call the default constructor
    - ☑ ~~Call the copy constructor~~
    - ☐ Call the assignment operator then the copy constructor

13. Which of the following statements about std::string in C++ is incorrect?
    - ☐ std::string objects automatically manage memory.
    - ☐ std::string can contain null characters ('\0') within the string.
    - ☑ ~~std::string concatenation always results in a new string object being created.~~
    - ☐ std::string provides random access to characters.

USC ID_____ Name_____

14. In a nested try block, if the inner catch handler gets executed, (assuming no further exceptions occur) then this will happen next:
    - ☐ Program execution halts immediately.
    - ☐ The outer catch handler will also be executed.
    - ☐ The compiler will jump to the outer catch handler and then execute the remaining executable statements of main().
    - ☑ ~~The compiler will execute the remaining executable statements of the outer try block and then proceed to the code immediately following the outer catch block(s).~~

15. References are useful when we want to _____.
    - ☑ ~~use classic pass-by-reference~~
    - ☐ use dynamic memory
    - ☐ implement linked lists
    - ☐ interface with C-string code

USC ID_____ Name_____

## Question 3 (20 points) General fill in the blank

1. Fill in the blank for this assignment operator:

   _____**Object&**_ operator=(const Object& o){ . . .

2. Suppose we want to know how much memory a doubly-linked list holding integers (type int) is taking up for all of the nodes and the head/tail pointers in the list class. Assume Item* pointers are size 8 and N is the number of nodes in the list, write an expression for the total amount of memory taken up by the linked list:

   _____8+8+(8+8+4)*N_____

3. What step is missing from this list of steps for reading from a file:
   a. Open the file
   b. _____Check .fail() on the ifstream object_____
   c. Read data from the file
   d. Close the file

4. Below is the code for a recursive function that will calculate the sum of the numbers from 1 to N (you may assume N > 0). Fill in the recursive step:

```
int triangle_number(int n){
      if(n == 1){
          return 1;
      }
      return _____n + triangle_number(n-1)_____;
}
```

5. Consider the snippet below, then write the required delete statements so that the code would execute with no memory leaks. Note: there are 4 blanks, there may be anywhere from 0 to 4 delete statements required.

```
int main() {
      vector<char *>* s = new vector<char *>[3];
      s[0].push_back(new char);
      s[1].push_back(new char[10]);
      s[2].push_back((char *) "hello");
}
```

`delete s[0][0]` and `delete [] s[1][0]` and `delete [] s`

_____ _____

_____ _____

USC ID_____ Name_____

6. When implementing a linked list, you usually need to declare (i.e. create a definition of) two object types, what are they?

   _____`struct Item` and `class List` or similar._____

7. When accessing member variables of a class, if you access an object through a pointer, you should use the ___->_____ operator.

8. In a recursive function, the _____base case_____ implements the necessary condition of stopping the recursive call.

9. While arrays, vectors and deques can all be used to implement a list, the _____run time complexity_____ of the different operations (e.g. access by index, insert, removal) can help us choose which one is appropriate for a specific algorithm.

10. Marking a reference parameter to a function as "const" means that the function _____can not change_____ that parameter.

11. Marking a member function as "const" means that member function will ____not change_____ the data members of an instance.

12. When implementing an operator overload for a binary operator like "+" the function needs to return a _____new object by value_____, not one of the input operands.

13. What must we never do after calling delete on a pointer?
    _____reuse_____

14. One aspect of polymorphism we discussed in lecture is being able to call the member function of a _____derived_____ class through a pointer of a _____base_____ class type. Or similar

15. Technically destructors can throw exceptions, however what happens if an exception is thrown during the handling of an exception from a destructor?

    _____program crashes_____

16. When choosing between composition vs. inheritance, if reusing an object is the primary goal we should choose _____composition_____.

17. If you see an object (Object obj) used in the following way while(obj), what does this imply about the definition and implementation of the Object class?

USC ID_____ Name_____

_____Object has an operator bool() defined_____

18. If a derived class uses dynamic memory, we need to ensure that the _____destructor_____ of the base class is marked virtual.

19. ____head_____ recursion can let us operate on a singly-linked list in reverse order (starting from the tail) in O(N) time.

20. Simple recursion is often an alternative to _____iteration_____, i.e. doing something to every item in a list.

## Question 4 (3 points) Images

This section will be replaced for the 2026 Final

## Question 5 (5 points) Streams and parsing - Fill in the blank

Assume you are given a file whose name is "data.txt" and whose contents are shown below. Show what the program below will print when executed. Assume all required includes.

```
int main() {
    ifstream fin("data.txt");
    int m, n, total = 0;
    string line;
    char c;

    fin >> c;
    cout << c << endl;
    // 1st output

    getline(fin, line, ';');
    fin >> m >> n;
    cout << m << endl;
    // 2nd output

    getline(fin, line);
    getline(fin, line);
    cout << line.length() << endl;
    // 3rd output

    stringstream ss(line);
    for (int i = 0; i < n; i++) {
        int temp;
        ss >> temp;
        if (temp > 20)
            total += temp;
    }

    cout << total << endl;
    // 4th output
    getline(fin, line);
    c = fin.get();
    if(fin.fail())
        { cout << "fail" << endl; }
        // 5a output
    else
        { cout << c << endl; }
        // 5b output

    return 0;
}
```

This is the content of “data.txt”

```
# This is a comment
4; 12 3;
20 30 40 50
```

USC ID_____ Name_____

```
60
```

What is the output of this program for the 5 cout lines that will run:

1. _____#_____
2. _____12_____
3. _____11_____
4. _____70_____
5. _____fail_____

## Question 6 (8 points) Recursion Tracing - Fill in the blank

```
#include <iostream>
#include <string>
using namespace std;

int f1(int m, int cnt)
{
      cout << "m : " << m << endl;
      if(cnt > 2) {
            m += 1;
            return m;
      }
      else {
            m = m*2;
            cnt += 1;
            return f1(m, cnt);
      }
}

int main()
{
      int n;
      cin >> n;
```

USC ID_____ Name_____

```
    n = f1(n, 0);
    cout << "n : " << n << endl;
    return 0;
}
```

If we run this program and enter '4', what would the program print (there may be more lines than you need):

1. _____m:4_____
2. _____m:8_____
3. _____m:16_____
4. _____m:32_____
5. _____n:33_____
6. _____
7. _____

## Question 7 (8 points) Recursion Tracing - Fill in the blank

The following program contains two recursive functions, `f5` and `f6`, each defined by different conditions. Your task is to determine the exact values printed by the program. Carefully trace each recursive call.

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

int f5(int x) {
    if (x == 0)
        return 1;
    else if (x < 0)
        return f5(-x);
    else if (x == 1)
        return 2;
    else
        return x * f5(x - 2);
}

int f6(int n) {
    if (n <= 1)
        return n;
    if (n % 2 == 0)
        return f6(n / 2);
    else
        return 1 + f6(n - 1);
}

int main() {
    cout << f5(-4) << endl;
    cout << f5(5) << endl;
    cout << f5(6) << endl;
    cout << f6(10) << endl;
    cout << f6(27) << endl;
    return 0;
```

USC ID_____ Name_____

}

Answer for the 5 cout statements:

1. _____8_____
2. _____30_____
3. _____48_____
4. _____2_____
5. _____4_____

USC ID_____ Name_____

## Question 8 (6 points) Inheritance/Polymorphism - Fill in the blank

Consider the following code. Write down the output from main. If you don't think a particular line of code will compile, you MUST cross it out.

```
class J {
public:
      virtual void f1() {
            cout << "J1" << endl;
      }
      void f2() {
            cout << "J2" << endl;
      }
      virtual void f3() {
            cout << "J3" << endl;
      }
};

class K : public J {
public:
      virtual void f1() {
            cout << "K1" << endl;
      }
      virtual void f2() {
            cout << "K2" << endl;
      }
};

class M : public J {
public:
      void f2() {
            J::f2();
            cout << "M2" << endl;
      }
      virtual void f4() {
            M::f2();
            cout << "M4" << endl;
      }
};

class P : public M {
public:
      void f1() {
            cout << "P1" << endl;
      }
      virtual void f2() {
            cout << "P2" << endl;
            M::f4();
      }
};

int main()
{
      J* var1 = new K;
      var1->f1();
      var1->f2();

      M* var3 = new P;
      var3->f1();
      var3->f2();
      var3->f3();
}
```

Output (hint: there are 6 lines generated):

_____K1_____

_____J2_____

_____P1_____

_____J2_____

_____M2_____

_____J3_____

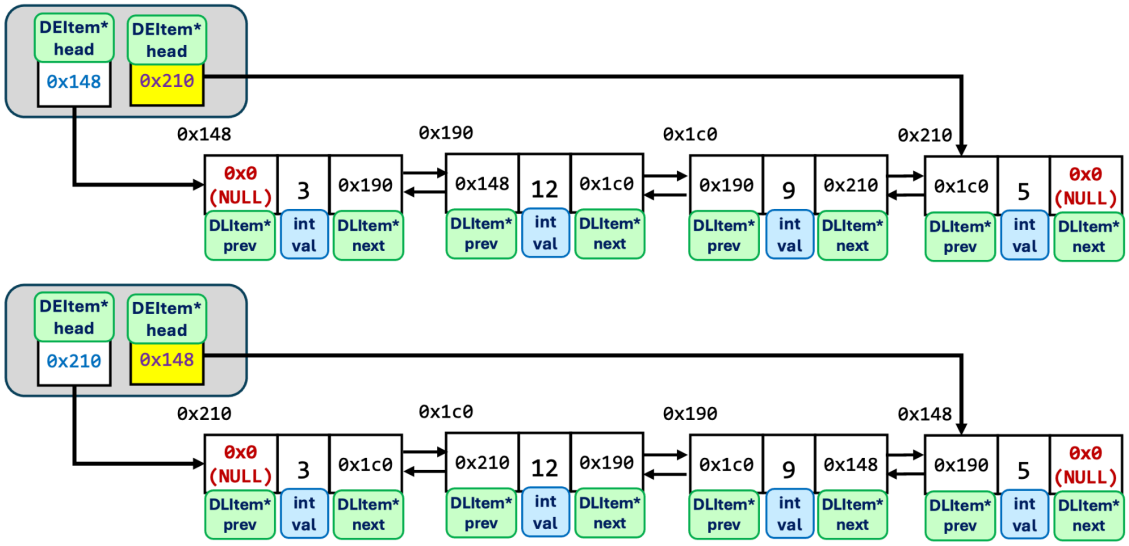
_____

_____

USC ID_____ Name_____

## Question 9 (10 points) Linked List - Fill in the blank

For this linked list problem you will be implementing an in-place reverse function. The idea is simple, shouldn't we be able to reverse a doubly-linked list simply by swapping the next and prev pointers of each node, while also swapping the head and tail pointers? Look at the two lists below, clearly the answer is yes!




The algorithm is:

- If the list is size 0 or 1, do nothing
- Otherwise, step through the list swapping next and prev on each node
- Finally, swap head and tail

```
void List::reverse()
{
    if( !head || !head->next)
return;
 //blank 1a + 1b

    DEItem* temp = head;  //blank 2

    while(temp != NULL)  //blank 3

    swap(temp->next, temp->prev);
//blank 4
    {
      temp = temp->prev;//blank 5

      swap(head, tail);//blank 6
    }
    _____//blank 6
}
```

For any swap operation, use the builtin swap() function.

Blank 1a/1b: check if the list is empty or size 1, if so, do nothing. For this list there is no size() or size_ data member.

Blank 2: Whenever we traverse a linked list, we need to do something with the head pointer. Do that here. You will need this pointer inside the loop.

Blank 3: What type of loop and loop condition will we need here to traverse the entire list?

Blank 4: swap the current nodes next and prev (use swap())

USC ID_____ Name_____

Blank 5: update the current node to the next node. Hint: you're doing this *after* you swapped prev and next.

Blank 6: swap head and tail

(left blank for scratch)