CS 103: Introduction to Programming
Fall 2016 - Written Final Exam
12/10/16, 11AM –1PM

Name: _____

USC Email Address: _____

Lecture (Circle One):
Redekopp 9:30 TTh | Goodney: 2 MW | 11:00 TTh | 12:30 TTh

Page	Your score	Max score
2		8
3		10
4		4
5		7
6		5
7		7
8		5
9		4
Total		**50**

Note 1: You need NOT worry about #include or 'using namespace' statements.
Note 2: The last page is blank and for scratch work. You MUST turn it in with your exam.

1. (3 pts) Consider the following recursive function:

```
int d(int n, int m){
  cout << n << " " << m << endl;
  if( n < m ){
    return 0;
  }
  else {
    return d(n-m, m) + 1;
  }
}
```

   a. What is the return value for the call: `d(10, 3)`? _____

   b. What is printed to cout for the call: `d(7, 2)`?

2. (4 pts). Consider the following recursive function.

```
bool fn(char *s, int len) {
    cout << "len =" << len << endl;
    if (len <=1 ) return true;
    else return( (s[0] == s[len-1]) && fn(s+1, len-2) );
}// fn()
```

   a. What is the return value for the call: `fn("abc", 3)`? _____

   b. What is printed to cout for the call: `fn("noon",4)` ?

   c. Briefly explain what this function does at a high level (i.e. what kind of inputs will cause the result to be 'true' vs. 'false')?

3. (1 pt) When it comes to accessing members and methods, when would you use the dot (.) notation, and when would you use the arrow (->) notation?

4. (4 pts) For each operation below indicate if it will take **constant** time (i.e. O(1)) or **linear** time (O(n)) where n is the number of items in the array/list/vector.

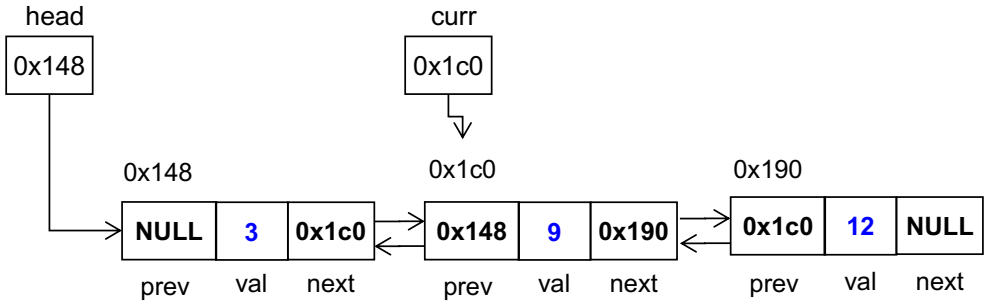
   a) Inserting to the back of a vector **constant or linear**

   b) Remove from the front of a deque **constant or linear**

   c) Inserting to the back of a deque **constant or linear**

   d) Remove from the front of a linked list **constant or linear**

5. (3 pts) The following shows a doubly linked list with three nodes. Assuming this configuration with 3 nodes, suppose we want to delete the node pointed to by 'curr' (i.e. node w/ value 9) and leave the linked list in a correct state afterwards. Circle which code operations are correct. More than 1 may be correct, circle all that apply.



Option A Works: Yes / No	Option B Works: Yes / No	Option C Works: Yes / No

curr->prev->next = curr->next;
curr->next->prev = curr->prev;
delete curr;
	Node* temp = curr->next;
	temp->prev->prev->next = temp;
	temp->prev = curr->prev;
	delete curr;
		Node* temp = curr->prev;
		curr->prev = curr->next;
		curr->next = temp;
		delete curr;

6. (1 pt) Using the original linked list shown in the previous problem and the **'curr'** pointer as shown, write a single cout statement that would print the value 12 to the screen. (Do **NOT** just write `'cout << 12;'` You should access the linked list node and get 12 to be printed).

7. (2 pts) Given an array with n elements, what is the best run-time complexity that can be achieved for an algorithm (with no prior knowledge) to check if a value does **NOT** appear in the array:

   a. If the array **is NOT** sorted? O(_____)

   b. If the array **IS** sorted? O(_____)

8. (1 pt.) What is the big-O runtime of the following code: O(_____)

```
int i=1;
//assume some large N
while(i < N)
{
    i *= 2;
}
```

9. (2 pts) What would the following program output?

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

int main() {
 std::deque <std::string> names;
 names.push_front("John");
 names.push_back("Jane");
 names.push_front("Joan");
 while (names.size() > 0) {
   std::cout << names.front() << std::endl;
   names.pop_front();
 }
 return 0;
}// main()
```

Output:

10. (1 pt.) Given a text file ("dat.txt") whose contents are `4 5 `, (i.e. spaces after 4 and 5 but no newline at the end…just the EOF (End-of-File). What will the following code print?

```
ifstream ifile("dat.txt");
int temp, cnt = 0;
while( ! ifile.fail()){
   ifile >> temp;
   cnt++;
}
cout << cnt << endl;
```

Output:

11. (4 pts.) Consider the following 5x5 maze (with rows R0..R4 and columns C0..C4), where S denotes Start, and F denotes Finish. #s are walls and .s (dots) are open spaces. Solving the maze consists of starting at S and finding a path to F. Assume we perform a **breadth-first search** where at each location we explore neighbors in the order: **Up, Left, Down, Right** (i.e. in counter-clockwise order). What squares would be placed in the search queue and in what order? List them all on the line below. We've started it for you. Stop when you encounter the F square as a neighbor for the first time.

	C0	C1	C2	C3	C4
R0	#	.	F	.	#
R1	.	.	.	.	#
R2	.	#	#	S	.
R3	.	.	.	.	#
R4	#	#	.	#	#

(R2,C3), _____

_____

12. (3 pts) The following program shows a function, f, and its invocation (call), using **pointer** notation. Re-write the program on the right side to instead use C++ reference-based syntax instead.

Pointer-based Implementation	**Equivalent C++ Reference (i.e. &) argument-passing implementation**

`#include <iostream>`
`using namespace std;`
 	
`void f(int *x){`
`  *x += 5;`
`}`
 	
`int main() {`
`  int k=15;`
`  f(&k);`
`  cout << k << endl; // outputs 20`
`  return 0;`
`}`	

13. (5 pts) Assume a file whose name is "file.txt" and whose contents are shown below. Show what the program below will print when executed.

File contents of “file.txt”:

```
5 3 2
6 5 4 3 2 1
7 8
```

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

int main()
{
  ifstream f("file.txt");
  int t, x, y;
  string z;
  f >> x >> y;
  cout << "L1: " << y << endl;
  getline(f, z);
  getline(f, z);
  cout << "L2: " << z << endl;
  stringstream ss(z);
  while(y > 0){
      ss >> t;
      x += t;  y--;
      cout << x << endl;
  }

}
```

Output:

14. (7 pts) Examine the code below. For the lines A1-A4 indicate what will be printed. For lines B1-B3 indicate whether they will compile or not.

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

class ABC {
public:
  ABC() { a = 7; b = 3.0; }
  ABC(int mya, double myb) { a = mya; b = myb; }
  int a;
  double get_b() { return b; }
private:
  double b;
};

class XYZ {
public:
  XYZ(ABC mya) { y = mya; x = 5; }
  int get_val() { x++; return doWork(); }
private:
  int doWork() { return x + y.a; };
  int x;
  ABC y;
};

int main()
{
  ABC a1;
  ABC a2(8, 12.5);
  XYZ x1(a1);

  cout << x1.get_val() << endl;  // Line A1
  --a1.a; cout << a1.a << endl;  // Line A2
  cout << x1.get_val() << endl;  // Line A3
  cout << a2.get_b() << endl;    // Line A4

  x1.y = a2;                     // Line B1
  cout << a1.b << endl;          // Line B2
  cout << x1.doWork() << endl;   // Line B3
  return 0;
}
```

Enter your answers here:

Line A1: Output: _____

Line A2: Output: _____

Line A3: Output: _____

Line A4: Output: _____

Line B1: Compiles (Yes / No)

Line B2: Compiles (Yes / No)

Line B3: Compiles (Yes / No)

15. (5 pts) Complete all the blanks in the function f1() below such that it implements the operations described in the comments. You may only fill in the blanks and not add or alter other code. Also complete main().

```
// f1 copies the last *half* of vec's values into a new array pointed to 'arr'.
//  'arr' is garbage coming in and must be modified by this function so the
//  caller can then use the produced array.  The values copied from vec
//  should then be removed by vec. If there are an odd number of elements in
//  vec, leave the middle element in vec, and let arr have one less element
//  than vec
int f1(_____________________ vec, int **arr)
{
  int arrlen = _______________________________;
  _____________ = new int[arrlen];

  for(int i = _____________________; i >= ________; i--){
    (*arr)[i] = vec.back();
    vec._____________________________;
  }
  return arrlen;
}

int main()
{
  vector<int> vec1(3);   vec1[0] = 3; vec1[1] = 4; vec1[2] = 5;
  vector<int> vec2(2);   vec2[0] = 3; vec2[1] = 4;

  int *array1, *array2;
  int n1, n2;

  n1 = f1(vec1, _____________); // show how to pass array1
  // Should return with vec1 = {3, 4} and array1 = {5}, n1 = 1

  n2 = f1(vec2, _____________); // show how to pass array2
  // Should return with vec2 = {3} and array2 = {4}, n2 = 1

  // Show cleanup code, if any


  _________________________________

  _________________________________
  return 0;
}
```

16. (4 pts) Consider the following recursive function and show what will be printed to the screen (assume appropriate #include's, prototypes, and using statements).

```
void mystery(vector<string>& s1, vector<string>& s2)
{
  if(s1.size() > 0){
    mystery_help(s1, s2, 1, 0, s2[0]);
  }
  if(s2.size() > 0){
    mystery_help(s1, s2, 0, 1, s1[0]);
} }

void mystery_help(vector<string>& s1, vector<string>& s2,
                  int x, int y, string p)
{
  if(x == s1.size()-1 && y == s2.size()-1){
    cout << p << endl;
  }
  else {
    if(x < s1.size()-1){
      mystery_help(s1, s2, x+1, y, p + "->" + s2[y]);
    }
    if(y < s2.size()-1){
      mystery_help(s1, s2, x, y+1, p + "->" + s1[x]);
    }
} }

int main()
{
  vector<string> s1, s2;
  s1.push_back("A"); s1.push_back("B"); s1.push_back("C");
  s2.push_back("1"); s2.push_back("2");
  mystery(s1, s2);
  return 0;
}
```

Output:

This page is for scratch work. You MUST turn it in with your exam…