CS 103: Introduction to Programming
Fall 2017 - Written Final Exam
12/11/16, 4:30PM –6:30PM

Name: _____

USC Email Address: _____

Lecture (Circle One):

Redekopp 2:00 TTh | Goodney: 2 MW | 9:30 TTh | 11:00 TTh

Complete the Information Above for 1 point of credit.

Page	Your score	Max score
1		1
2		7
3		8
4		4
5		7
6		5
7		7
8		6
9		5
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. (3 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)` ?

3. (1 pt) The following code fragment doesn't compile. Explain why in a short sentence: _____

```
struct D { int x; double y;};

int main()
{
      D* my_d = new D;
      my_d.x = 10;
      my_d.y = 3.14;
}
```

4. (2 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.

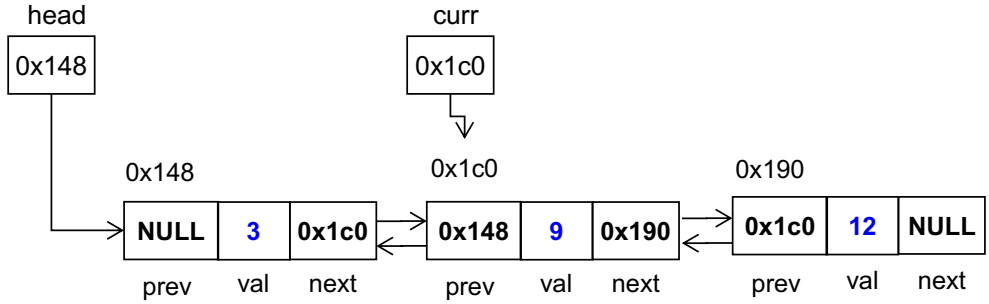
   a) Inserting to the back of a vector **constant or linear**

   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) Billy Bruin finishes his CS degree and gets a job as a C++ programmer. As part of a project he writes the following code fragment:

```
vector<Object> v;
for(i=0;i<N;i++)
{
      Object o = objectFactory.getNextObject();
      v.insert(v.begin(), o);
}
```

Billy was proud of this code, however during testing when N is large his code is very slow. You proposed the following rewrite:

```
vector<Object> v;
for(i=0;i<N;i++)
{
      Object o = objectFactory.getNextObject();
      v.push_back(o);
}
reverse(v); // Reverses the vector contents
```

Explain in one short sentence using your knowledge of vectors why your solution is much faster:

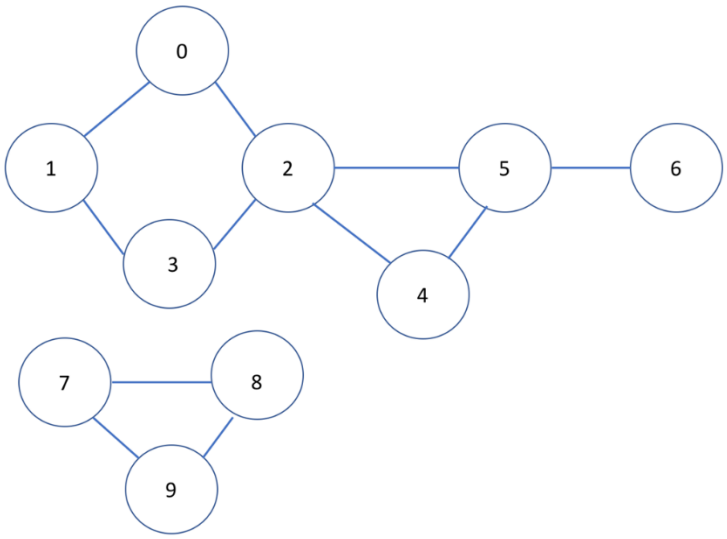
10. (1 pt.) Given a text file ("dat.txt") whose contents are `4 5 `, (i.e. spaces after 4 and 5 but no newline at the end…just the EOF (End-of-File). What will the following code print?

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

Output:

11. (4 pts.) Consider the following graph. We are going to do a **breath-first search** on the graph starting at **node 3**. Assume the neighbors of a particular node are stored **sorted by node ID**. Nodes are only discovered once (assume there is some sort of visited flag or list). You can pretend this is a "friends" graph like PA5/6. On the line below, fill-in the order in which the nodes are discovered and put into the queue (we started it for you):

queue: 3, _____



12. (3 pts) The following program shows a function, f, and its invocation (call), using **pointer** notation for passing the parameter. 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 you are given 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. (6 pts) Consider the following program that should read in 1024 words (strings) from a file into an array, and returns the array to main which then outputs 512 strings. The 512 strings should be the result of concatenating the *first and last*, *second with second to last*, *third with the third to last*, etc. Fill in the indicated blanks below (return type, array allocation, `words` declaration, cout code, and cleanup).

```
#include <iostream>
#include <fstream>

using namespace std;

// This function reads 1024 words from the given file
// fill in the return type below

____________  readDict(char* f) {
      ifstream inFile(f);

      // allocate an array 's' of 1024 strings

      string_____________________________;
      for(int i=0; i<1024; i++) {
        inFile >> s[i];
      }
      return s;
}// readDict()

int main() {
      // Indicate the type of words below

      _________ words = readDict("dict.txt");

      for(int k = 0; k < 512; k++){

            cout << words[k] ____________________________ << endl;
      }

      // now we're done working with the words

      // Show any cleanup code you need here (if any)




      return 0;
}
```

16. (5 pts) Consider the following program utilizing recursion and show what will be printed to the screen.

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


void myst(vector<string>& sol, int n, string pre)
{
  if(n > 0){
    for(unsigned k=0; k < pre.size()+1; k++){
      char c = 'a' + k;
      myst(sol, n-1, pre + c);
    }

  }
  else {
    sol.push_back( pre  );
  }
}

int main()
{
  vector<string> items;
  myst(items, 3, "");
  for(unsigned j = 0; j < items.size(); j++){
    cout << items[j] << endl;
  }
  return 0;
}
```

Output:

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