ANSWER KEY

Polytechnic Tutoring Center

Exam 1 Review - CS 1124, Spring 2013

Disclaimer: This mock exam is only for practice. It was made by tutors in the Polytechnic Tutoring Center and is not representative of the actual exam given by the CS Department.

1. Declare:
   a. An unchanging pointer to an int — `int* const ptr;`
   b. A pointer to an unchanging int — `const int* ptr;`
   c. An unchanging pointer to an unchanging int — `const int* const ptr;`

2. What will the following code result in?

```
int main(){
        int i = 7;
        const int * ip = &i;
        cout << i++ << endl;
}
```

   a. 7
   b. 8
   c. Runtime error
   d. Compilation error

3. True / False? – Circle answers that are true

   a. When passing a pointer by value, you can't change the original pointer or its contents.

   b. **Creating an object of any class is called: instantiation.**

   c. If class A has a private data member: int x, and Class B derives from Class A, Class B can directly access and change the data member int x.

   d. **To get the size of a vector, you use the method size()**

4. What will the following code result in?

```
class A {
      int foo;
};

class B : public A{
      int bar;
};

int main(){
      B myB;
      A myA = myB;
}
```

   a. Compilation error
   b. Slicing problem
   c. Both a and b
   d. None of the above

5. Write a function, called readPeople, that takes an ifstream and vector of Person pointers. It will use the ifstream to read from a file containing information about people. Each line contains a person's name and number separated by whitespace. The function will then use the person's information to create a Person object on the heap. Finally, the function should store the Person object's pointer in the vector.
   The Person struct and the sample file are given below.

```
struct Person
{
      string name, number;
};

Sample file:

John_Sterling
1-111-222-3333

Linus_Torvalds
1-888-555-9999
```

```
void readPeople(ifstream& ifs, vector<Person*>& v)
{
      string name, number;
      while (ifs >> name >> number)
      {
            Person* temp = new Person;
            temp->name = name;
            temp->number = number;
            v.push_back(temp);
      }
}
```

6. Write a function to delete all the data created in the previous problem.

```
void deleteVector(vector<Person*>& v)
{
	for (size_t i = 0; i < v.size(); i++)
	{
		delete v[i];
	}
	v.clear();
}
```

7. Write a function to search the vector from the problems above. The function should take the vector of Person pointers and the phone number of the person, and return their index in the vector.

```
int SearchByNum(const vector<Person*>& v, const string& num)
{
	for (size_t i = 0; i < v.size(); i++)
	{
		if (v[i]->number == num){
			return i;
		}
	}
	return -1;
}
```

8. Write a function that increments all elements of a passed in vector of ints. You must use a ranged-based for loop to increment the elements.

```
void incrementElements(vector<int>& v){
	for(int& x:v)
		x++;
}
```

9. Write a class Employee to model the employees in a company PolyCo.
   - In PolyCo, each employee has a name, can have one boss and zero or more sub-employees.
   - The CEO of PolyCo doesn't have a boss, of course, but every other employee does.
   - When employees are created, there are the following two possibilities:
     - They can be told who their boss is right away.
     - They can be created without a boss.
   - Employees should be able to be added and removed from the list of sub-employees at any time in the future. Provide support for this. **Think about the fact that employees in the team need to know who their boss is, and the boss needs to know who's in his team.**
   - Write a display function that will give the output as shown.
   - **Important: Only write the class Employee!**
   - **Assume that there will be no duplicate employees added to a boss employee.**
   - Here's a sample main and the sample output it produces.

```
int main()
{
	Employee sterling("Professor Sterling");
	Employee yan("Yan", &sterling);
	Employee jeremy("Jeremy");
	Employee mike("Mike");
	yan.addToTeam(jeremy);
	yan.addToTeam(mike);
	yan.display();
	cout << endl;
	sterling.display();
	cout << endl;
}
```

Output:

```
Name: Yan
Boss: Professor Sterling
Team...
	Jeremy
	Mike

Name: Professor Sterling
Boss: I am the boss.
Team...
	Yan
```

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

class Employee
{
	string name;
	Employee* boss;
	vector<Employee*> myTeam;
public:
	//Constructor
	Employee(const string& name, Employee* boss = NULL) : name(name),
		boss(boss)
	{
		if(boss != NULL){
			boss->myTeam.push_back(this);
		}
	}

	void addToTeam(Employee& newGuy)
```

```
    {
        if (newGuy.boss != NULL)
        {
            newGuy.boss->removeFromTeam(newGuy);
        }
        newGuy.boss = this;
        myTeam.push_back(&newGuy);
    }

    void removeFromTeam(Employee& emp)
    {
        for (size_t i = 0; i < myTeam.size(); i++)
        {
            if (myTeam[i] == &emp)
            {
                emp.boss = NULL;
                myTeam[i] = myTeam[myTeam.size()-1];
                myTeam.pop_back();
                break;
            }
        }
    }

    void display() const
    {
        cout << "Name: " << name << endl;
        cout << "Boss: ";
        if (boss != NULL)
        {
            cout << boss->name << endl;
        }
        else
        {
            cout << "I am the boss." << endl;
        }
        cout << "Team..." << endl;
        for (size_t i = 0; i < myTeam.size(); i++)
        {
            cout << "\t" << myTeam[i]->name << endl;
        }
    }
};
```