**Name: Michael Steve Duarte**   
  
***Investigative Section:***  
  
**A:1) List your top two languages in descending order of fluency.**

Java and C++  
  
**A:2) Please provide a link to some source code you have written which you are proud of (if available).**

My github account: https://github.com/MiichaelD

The proudest code I have participated for is a graphics engine written in C++ from scratch for an OpenGL ES 2.0 course while working in Gameloft, I have no link for it because I left it on Gameloft private repository.

The code I have available and I feel proud is the one for the game Color Flooded: <https://github.com/MiichaelD/Colorized_Android> because I made it completely from scratch to publishing and also because I put knowledge like security (encrypting and decrypting, using secure data types to prevent hacking tools of changing values), Social networks (Leaderboard, Achievements and Sharing by Google and also FB sharing), Adding 3rd party libs for ads, etc. on practice.  
  
**A:3) If you have any software projects you have worked on with their own websites link them here (if available).**

Color Flooded (Invasión de Color):

<https://play.google.com/store/apps/details?id=com.webs.itmexicali.colorized>

Gangstar Vegas Paymium HD+:

<https://play.google.com/store/apps/details?id=com.gameloft.android.ANMP.GloftGGHM>

Ice Age Village Freemium HD+:

<https://play.google.com/store/apps/details?id=com.gameloft.android.ANMP.GloftIAHM>

Asphalt 7: Heat Paymium HD+:

<https://play.google.com/store/apps/details?id=com.gameloft.android.ANMP.GloftA7HM>

Candy Block Breaker for Tango:

<https://play.google.com/store/apps/details?id=com.gameloft.android.ANMP.GloftCTHM>

Real Gaming: BattleShock:

<https://play.google.com/store/apps/details?id=com.webs.itmexicali.rg.BattleShock>

Bluino for Android:

<https://play.google.com/store/apps/details?id=com.webs.itmexicali.bluino>

UNO & Friends HD+:

<https://play.google.com/store/apps/details?id=com.gameloft.android.ANMP.GloftUOHM>

***Test Section:***  
  
**B:1) Answer the following questions about the code snippet.**  
  
**a) List all of the problems you can see with this code and provide specific minimal changes which would address those problems.**

It has 2 problems:

1) the newBuffer function must receive the reference to the pointer foo, so it can initialize it and prevent a segmentation fault that was caused when trying to clean the foo's memory block using the memset function (because it was really not initializated).

so it should look something like this:

**void newBuffer(char\*\* outBuffer, size\_t sz) {**

**\*outBuffer = new char[sz];**

**}**

And in the main function call the newBuffer function like this:

**newBuffer(&foo, len);**

2) also there is a memory leak because when allocating memory using malloc or new operator the allocation happens in the heap memory and should be dellocated or deleted using the free/delete operator:

**delete[] foo;**

**b) OPTIONAL BONUS: Describe how RAII could help simplify this implementation if it were rewritten.**

A really easy modification to implement RAII idiom would be to change the variable foo from a pointer of chars and allocating it in the heap to an array of chars that will be freed when exiting the scope of the current method on stack:

**char foo[len];**

//char \*foo;

//newBuffer(&foo, len);

*#include <iostream>*  
*#include <string.h>*  
  
*using namespace std;*  
  
*void newBuffer(char\* outBuffer, size\_t sz) {*  
*outBuffer = new char[sz];*  
*}*  
  
*int main(void) {*  
*const char\* kung = "KUNG";*  
*char\* foo;*  
*size\_t len = strlen(kung);*  
  
*newBuffer(foo, len);*  
*memset(foo, 0, len+1);*  
*strncpy(foo, kung, len);*  
  
*cout << foo << endl;*  
*}*  
  
**B:2) Write a Car class.**  
**Assuming you are given a class “Engine” that has the “start()” method prototyped below, write a “Car” class with a “turnKey()” method that tells you whether the car started or not. The Car class should encapsulate an “Engine”.**  
  
**bool Engine::start();**  
  
#include <iostream>

class Car{

private:

Engine \*mEngine;

public:

Car(){

mEngine = new Engine();

}

~Car(){

if(mEngine != NULL)

delete mEngine;

}

bool turnKey(){

bool started = pEngine->start();

if(started){

cout << (started? "The engine started" : "The engine didn't start") << endl;

}

return started;

}

}  
  
  
**B:3) Describe precisely what happens in each step of this program and what the output will be. Assume that setString sets an instance member variable and printString prints it with no additional space or formatting.**  
  
1)*MyClass \*a = new MyClass();*  
2)*vector<MyClass> myVector;*  
3)*myVector.push\_back(\*a);*  
  
4)*MyClass b = myVector[0];*  
  
5)*a->setString(“Hello World”);*  
6)*b.setString(“Goodbye World”);*  
  
7)*a->printString();*  
8\_*b.printString();*

Description per line number:

1) Declare a pointer “a” and instantiate an object of type MyClass on the heap memory.

2) Create a vector of type MyClass.

3) Push at the end the value of the object a (a copy) to myVector.

4) Create an object in the stack called b and copy the value of the first position of myVector.

5) Set “Hello World” to the member string of the object located by the pointer a using its function setString.

6) Set “Goodbye World” to the member string of the object b using its function setString.

7) Prints the value of the member string of object a (“Hello World”);

8) Prints the value of the member string of object b (“Goodbye World”);

As note: the member string of myVector[0] was not modified. And the allocation made by the first line was never freed using “delete” operator.

**B:4) Given the following:**  
*friend ostream& operator<<(ostream& out, Obj& myObj);*  
**a) Describe what this line of code declares.**

This line of code overrides the << operator to specify how we want the instances of our class Obj to be printed using “cout”, it is friend to let the function access private data types within our class.

**b) Why do we return a reference to ostream?**

Because in that way we can combine insertions and concatenate more and more objects to be printed after our object.

I. e. if we didn't return the reference to ostream we couldnt add more printable objects (underlined): cout<< “Object :” << obj << “To concat this too” << endl;

**c) Why couldn’t we declare a member function like so? (ie: why pass in Obj at all?)**  
*ostream& operator<<(ostream& out);*  
  
Because we are specifying that this override needs an output stream followed by an object of our class.  
  
  
  
  
**B:5) Using the following code implement the following methods.  You may use a compiler to verify your results.**  
  
**a) Node::getNode returns a pointer to the first node in which Node::value matches the supplied argument "searchValue".  If includeThis is true then the first node's value is checked and the function may return *this*, if false, only a child node will be returned. Implement this method using a depth-first search.**  
  
**b) Node::getDepth returns an integer value indicating how many parents the current node has (a node does not have a parent if its parent pointer is set to 0).  Implement this method.**  
  
**c) OPTIONAL BONUS: Feel free to implement getNode as a breadth-first search.**  
  
*#include <vector>*  
  
*using namespace std;*  
  
*class Node {*  
 *Node\* parent;*  
 *vector< Node\* > children;*  
 *int value;*  
*public:*  
 *Node(int a\_value):value(a\_value),parent(0){}*  
 *~Node(){for(int i = 0;i < children.size();++i){delete children[i];}}*  
  
 *int getValue(){*  
 *return value;*  
 *}*  
 *Node\* addChild(const Node &child){*  
 *children.push\_back(new Node(child));*  
 *children.back()->parent = this;*  
 *return children.back();*  
 *}*  
 *Node\* getNode(int searchValue, bool includeThis = false);*  
 *int getDepth();*  
*};*

Attached on separated .cpp file.