Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 312, Spring 2015
Aater Suleman, Instructor
Owais Khan, Chirag Sakhuja, TAs
Exam 2, April 22, 2015

Name: Solution

Problem 1 (25 points):_______

Problem 2 (20 points):_______

Problem 3 (15 points):_______

Problem 4 (20 points):_______

Problem 5 (20 points):_______

Total (100 points):_____________

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

I will not cheat on this exam.

____
Signature

GOOD LUCK!

Name:____

Problem 1. (25 points):

The following code pertains to parts a and b below.

```
class Coordinate
{
    int32_t x;
    int32_t y;

public:
    Coordinate() { x = 0x0000dada; y = 0xdeadfeed; }
};

class Circle {
    int32_t radius;
    Coordinate center;

    float area() { return pi*radius*radius; }

public:
    Circle () { radius = 0x0000beef; }
};

int main() {
    Circle *c = new Circle; ← 4 bytes
    Circle d;               ← 12 bytes

    // <----- POINT A ----------- >
}
```

Part a. (5 points): How many bytes of memory are allocated on the stack and heap when the computer is executing the statement at POINT A. Assume that addresses are 32-bits and compiler does *not* add padding.

Bytes on heap: 12 Bytes

Bytes on stack: 16 Bytes

The problem is continued on the next page.

Name:____

Part b. (5 points): What are the contents of the stack at POINT A?

Notes:

- Each entry in the figure represents one byte.
- Little endian (little end first)
- Stack is growing towards the top.
- No compiler optimizations for alignment.
- The elemenets are laid out in memory in the order they are defined.
- Pointers are 32 bits.
- Stack is empty at start.
- The first new (first line in main) allocates heap memory at location 0x00014325.
- Struct and Class elements are always pushed in the order they are defined.

	xFFF0	xef
	xFFF1	xbe
	xFFF2	x00
	xFFF3	x00
	xFFF4	xda
	xFFF5	xda
	xFFF6	x00
	xFFF7	x00
	xFFF8	xed
	xFFF9	xfe
	xFFFA	xad
	xFFFB	xde
	xFFFC	x25
	xFFFD	x43
	xFFFE	x01
Stack pointer →	xFFFF	x00

Note: Class elements are pushed in class order!

Circle d

↑ Stack grows up

Circle *c

The problem is continued on the next page.

Name:______________________________________

Part c. (5 points): What is the major difference between a struct and a class in C++? Answer in less than 20 words. A TA was able to answer in 9 words.

A struct has public members by default, a class has ~~public~~ private members by default

Part d. (5 points): An Aggie creates a phone book using a hash table with five buckets. After inserting 1000 entries, she finds that the five buckets contains 7, 0, 711, 17, 265 entries each. What advice will you give to the Aggie? Circle one of the following.

A) Increase the number of buckets to 10.

[✓ B)] Change the hash function.

C) Reduce the number of buckets to 2.

Please explain your reasoning in less than 20 words.

Changing the hash function to be better would distribute the entries more uniformly

Part e. (5 points): Consider the following template defined in C++.

```
template <typename T>
class Foo {
    T bar;
    T zoo;
};
```

What is the output for the following code pieces?

* Correction: pointers are 32-bits

sizeof(Foo<int32_t>): 8 Bytes

sizeof(Foo<int64_t>): 16 Bytes

sizeof(Foo<int8_t *>): 8 Bytes

Name:____

Problem 2. (20 points):

Your job is to implement *Vec3D*, a C++ class that stores a 3-dimensional vector (i.e. it has three components). Note for this problem, we are referring to a vector by its mathematical definition and not the STL object `std::vector`. You will implement two operations for Vec3D: the dot product and scalar multiplication. Recall that given two vectors, the dot product computes a scalar value as follows.

$$\mathbf{X} \cdot \mathbf{Y} = x_1y_1 + x_2y_2 + x_3y_3$$

Also recall that given a vector and a scalar, scalar multiplication computes a new vector by multiplying the three individual components by the scalar.

Below we have defined the Vec3D class. Your job is to complete the implementation of the functions.

```
class Vec3D
{
public:
    float values[3];

    Vec3D(float x, float y, float z) {
        values[0] = x;
        values[1] = y;
        values[2] = z;
    }

    float operator*(Vec3D& vec);
    Vec3D operator*(float scalar);
    Vec3D operator*=(float scalar);
};
```

Hint: When the implementation of Vec3D is complete, we run the following code.

```
int main() {
    Vec3D a(1.0f, 1.0f, 1.0f);
    Vec3D b(2.0f, 2.0f, 2.0f);

    a = a * 2.0f;
    printf("%f %f %f\n", a.values[0], a.values[1]. a.values[2]);
    a *= 2.0f;
    printf("%f %f %f\n", a.values[0], a.values[1]. a.values[2]);
    printf("%f\n", a * b);
}
```

The output is shown below.

```
2.0 2.0 2.0
4.0 4.0 4.0
24.0
```

The problem is continued on the next page.

Name:____

Fill in the implementation of the following functions.

```
float Vec3D::operator*(Vec3D& vec) {
    // Your code begins here
    float ret = 0;
    ret += values[0] * vec.values[0];
    ret += values[1] * vec.values[1];
    ret += values[2] * vec.values[2];
    return ret;

    // Your code ends here
}

Vec3D Vec3D::operator*(float scalar) {
    // Your code begins here
    float x = values[0] * scalar;
    float y = values[1] * scalar;
    float z = values[2] * scalar;

    return Vec3D(x,y,z);


    // Your code ends here
}


/* NOTE: The *= operator modifies the values in the object and then
 *        returns the modified object.
 */
Vec3D Vec3D::operator*=(float scalar) {
    Vec3D temp = (*this) * scalar;

    // Your code begins here
    this->values[0] = temp.values[0];
    this->values[1] = temp.values[1];
    this->values[2] = temp.values[2];
    return (*this);
    // Your code ends here
}
```

Name:____

Problem 3. (15 points):

Recall that the Fibonacci sequence $(1, 1, 2, 3, 5, 8, 13, ...)$. is defined as

$$F_1 = 1$$
$$F_2 = 1$$
$$F_N = F_{N-1} + F_{N-2}$$

Your job is to compare three possible algorithms to compute the Nth Fibonacci.

```
int32_t iterativeFib(int32_t N)
{
     int32_t prev = 0;
     int32_t fib = 1;
     int32_t i;

     for(i = 2; i <= N; i++) {
          int temp = prev;
          prev = fib;
          fib = fib + temp;
     }

     return fib;
}
```

```
int32_t recursiveFib(int32_t N)
{
     if(N <= 0) return 0;
     if(N == 1) return 1;

     int32_t n1 = recursiveFib(N - 1);
     int32_t n2 = recursiveFib(N - 2);

     return n1 + n2;
}
```

Part a. (5 points): The complexity (Big O) of `iterativeFib` is $O(N)$. What is the complexity of `recursiveFib` in terms of N?
Hint: You can try plugging in increasing values of N and tracing the execution by hand to identify trends.

`recursiveFib` $O(2^N)$

Part b. (10 points): If you answered Part a correctly, you know that `recursiveFib` runs much slower than the iterative alrogithm. Your job is to complete the implementation of a new recusive algorithm, `recursiveFib2` with the same complexity as the iterative algorithm above, $O(N)$.
Note: To compute the Nth Fibonacci number, you can call `recursiveFib2(N, 0, 1)`.

```
int32_t recursiveFib2(int32_t N, int32_t prev, int32_t fib)
{
     if(N <= 0) return prev;
     if(N == 1) return fib;

     return recursiveFib2(N-1, fib, fib+prev);
}
```

refer to iterative solution

Name:____

Problem 4. (20 points):

Use the following code to answer the subsequent questions. Note that this code is similar to the code we studied during the lecture *with* a few changes.
Note1: You may find it useful to read the questions before spending time understanding the following code.
Note2: The problem is testing your understanding of access specifiers (private/protected/public).

```
#include<vector>
#include<string>
#include<iostream>
// Base class User
class User {

protected:
  int32_t userId;
  std::string username;

private:
  std::string password;

public:
  User(){ }

  User(int32_t userId, std::string username, std::string password){
    this->username = username;
    this-> userId = userId;
    this->password = password;
  }

  bool login(std::string typedPassword) {
    if (typedPassword == password){
      return true;
    }
    return false;
  }

  void print(){
    std::cout<<"Print in User called for "<<username.c_str()<<std::endl;
  }
};
```

The problem is continued on the next page.

Name:________________________________

```
//Derived class Student
class Student : private User {

protected:
  std::string major;

public:
  Student(int32_t userId, std::string username, std::string password,
                          std::string major) : User(userId, username, password) {
    this->major = major;
  }

  void print() {
    std::cout<<"Print in Student "<< this->username.c_str()<<" called."<<std::endl;
  }


  void printStudent(char* type) {

    if(strcmp(type,"name")==0){
        std::cout<<"Student name is: "<<this->username.c_str()<<std::endl;
    }
    else if(strcmp(type,"major")==0){
        std::cout<<"Student major is: "<<this->major.c_str()<<std::endl;
    }
    else{
        std::cout<<"Invalid input."<<std::endl;
    }
  }

};

//Derived class Instructor
class Instructor : public User {

  std::string level;

public:
  Instructor(int32_t userId, std::string username, std::string password,
                          std::string level) : User(userId, username, password) {
    this->level = level;
  }

  void printInstructor(char* type) {

    if(strcmp(type,"name")==0){
        std::cout<<"Instructor name is: "<<this->username.c_str()<<std::endl;
    }
    else if(strcmp(type,"level")==0){
        std::cout<<"Instructor level is: "<< this->level.c_str() << std::endl;
    }
    else{
        std::cout<<"Invalid input."<<std::endl;
    }
  }
};
```

The problem is continued on the next page.

Name:____

Answer the following subproblems using the above code. For each of the following subproblems

1. Indicate if the code will compile or not by circling the correct answer.
2. If the code will compile, then write down the output generated when the program is run in the box provided below. Otherwise, write down the reason why it does not not compile.

Part a. (5 points):

```
int main(){
    Student s(123, "student1", "1234", "ECE");
    Instructor *i = new Instructor(456, "instructor1", "password1", "adjunct");
    bool succ = s.login("1234");
    s.print();
    i->print();
    delete i;
    return 0;
}
```

COMPILE: YES / [✓ NO] (circle one answer).

s.login ("1234)") fails because login is inherited privately by Student

Part b. (5 points):

```
int main(){
    Student s(123, "student1", "1234", "ECE");
    Instructor *i = new Instructor(456, "instructor1", "password1", "adjunct");
    bool succ = i->login("password1");
    s.print();
    i->print();
    delete i;
    return 0;
}
```

COMPILE: [✓ YES] / NO (circle one answer).

Print in Student ~~called~~ student1 called.

Print in User called for instructor1

The problem is continued on the next page.

Name:______________________________________

Part c. (5 points):

```
int main(){
    Student s(123, "student1", "1234", "ECE");
    Instructor i(456, "instructor1", "password1", "adjunct");
    std::cout<<"Student Major is: "<<s.major.c_str()<<std::endl;
    return 0;
}
```

COMPILE: YES / [✓ NO] (circle one answer).

s.major fails because major is protected in student

Part d. (5 points):

```
int main(){
    Student s(123, "student1", "1234", "ECE");
    Instructor i = Instructor(456, "instructor1", "password1", "adjunct");
    s.printStudent("major");
    i.printInstructor("name");
    i.resetPassword("newPassword");
    return 0;
}
```

* Correction: reset Password declared publicly in User and has no output

COMPILE: [✓ YES] / NO (circle one answer).

Student major is; ECE
Instructor name is: instructor1

Name:______________________________

Problem 5. (20 points):

Please answer the questions on the next page about the following code. You may assume that there are no compiler optimizations enabled and there is no padding on the stack.

```
#include <iostream>
#include <cstring>

class MyString{

private:
    char *data;

public:
    MyString() { this->data = NULL; }

    MyString(char* newData){
        this->data = new char[strlen(newData) + 1];
        strcpy(this->data, newData);
    }

    // capitalize function works correctly.
    void capitalize(){
        for(int i=0; i<strlen(this->data); i++){
                this->data[i]&=~0x20;
            }
    }

    char* c_str(){ return this->data; }

    ~MyString(){
        if(this->data){
            delete[] this->data;
            this->data = NULL;
        }
    }
};


int main(void){
    MyString a("Circle");
    MyString b;
    b = a;
    b.capitalize();
    std::cout << a.c_str() << std::endl;
    std::cout << b.c_str() << std::endl;

    return 0;
}
```

The problem is continued on the next page.

Name:____

Part a. (5 points): What is printed when the program is run? Please write it down in the box below.

```
CIRCLE
CIRCLE
```

Part b. (5 points): Please explain in less than 15 words why this output is different from the expected output.

When running b=a, the pointer is copied, so a and b point to the same string on the heap.

Part c. (10 points): Write an operator function which needs to be added to the String class so that it produces the expected output. Do not change the main function.

```
// Complete the function name and specify the required arguments

void  operator = (MyString& str)                {

    // Your code begins here

        if(data != NULL)
              delete[] data;
        data= new char[strlen(str.data)+1];
        Strcpy(data, str.data);



    // Your code ends here
}
```