Ryerson University
Department of Electrical & Computer Engineering
COE808

Midterm Examination **March 4, 2015**

Name:_____ **ID #: _____**

Time: 75 mins

Q1. Let the function fun be defined as (5 marks)

```
int fun(int *k) {
*k += 4;
return 3*(*k) –1;
}
```

Suppose fun is used in a program as follows:

```
void main() {                              i: 10
int i = 10, j=10, sum1, sum2;              k ↗
sum1 = (i/2) + fun (&i);
sum2 = fun(&j) + (j / 2);                  j: 10
}                                          k ↗
```

What are the values of sum1 and sum2
a. if the operands in the expressions are evaluated left to right?
b. if the operands in the expression are evaluated right to left?

Answer:
(a) (left -> right) `sum1` is 46; `sum2` is 48 **(1 mark each)**
(b) (right -> left) `sum1` is 48; `sum2` is 46 **(1.5 mark each)**

Q2. For each of the following programs, indicate each of the dangling pointer and lost heap-dynamic variable (memory leak) errors. Note that there may be 0, 1 or more defects in each category for each program.

Assume that the language does not deal with dangling pointers or lost heap-dynamic variable (e.g., through garbage collection). (10 marks)

(1) Dangling pointers (dangerous)

-A pointer points to a heap-dynamic variable that has been de-allocated

(2) Lost heap-dynamic variable

- An allocated heap-dynamic variable that is no longer accessible to the user program (often called *garbage*)

```
void main(void){
int *a, *b;                    b ⟶ 4
b = new int;
*b = 4;                        a ⟶ 4
a = new int;
*a = *b;
}
```

No memory leaks or dangling pointers. (2.5 each)

```
void main(void){
int *x;
   for (int i = 0; i <= 50; i++){     x ⟶  0
      x = new int;                         1
      *x = i;
   }
}
```

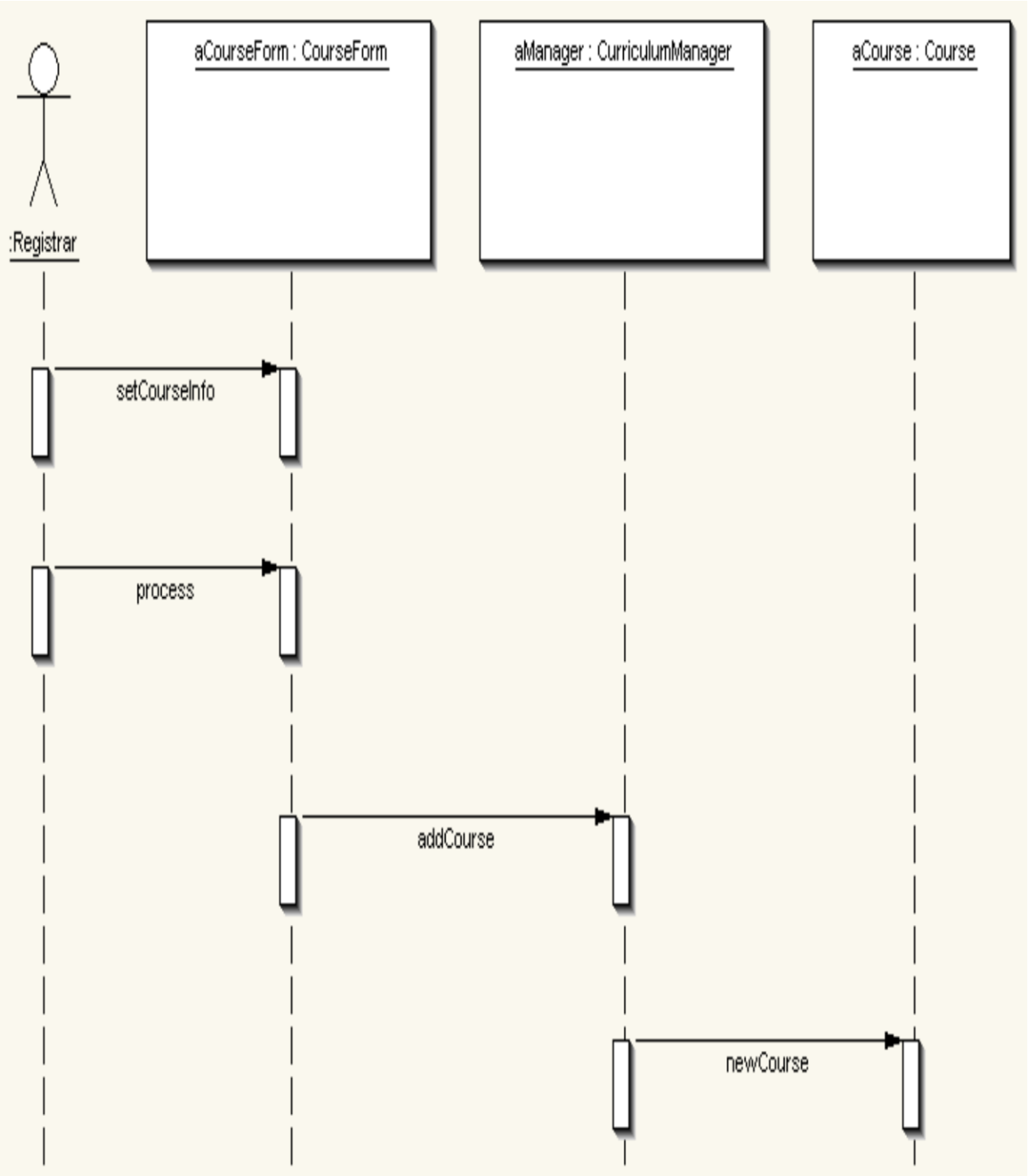
Memory leak: There is an allocation that occurs each time through the loop. Each time new memory is allocated, the old memory that x pointed to is effectively leaked.
No dangling pointer.
(5 marks or 0)

Q3. While designing a University Course Registration System, suppose the software architect came up with three classes: **CourseForm**, **CurriculumManager** and **Course**. Let the registrar of the university be an actor who interacts with the Course Registration System. Consider the following "Add a new course" scenario:

1. The registrar interacts with the system by invoking the method *setCourseInfo* on an existing object ***aCourseForm*** of the class CourseForm.
2. The registrar then interacts with the system again by invoking the method *process* on the object ***aCourseForm***.
3. Next the object ***aCourseForm*** invokes the methods *addCourse* on an existing object ***aManager*** of the class CurriculumManager.
4. The object ***aManager***, in turn, invokes the method *newCourse* on an existing object **aCourse** of the class Course.

Draw the sequence diagram for the above scenario. (5 marks)

Answer:

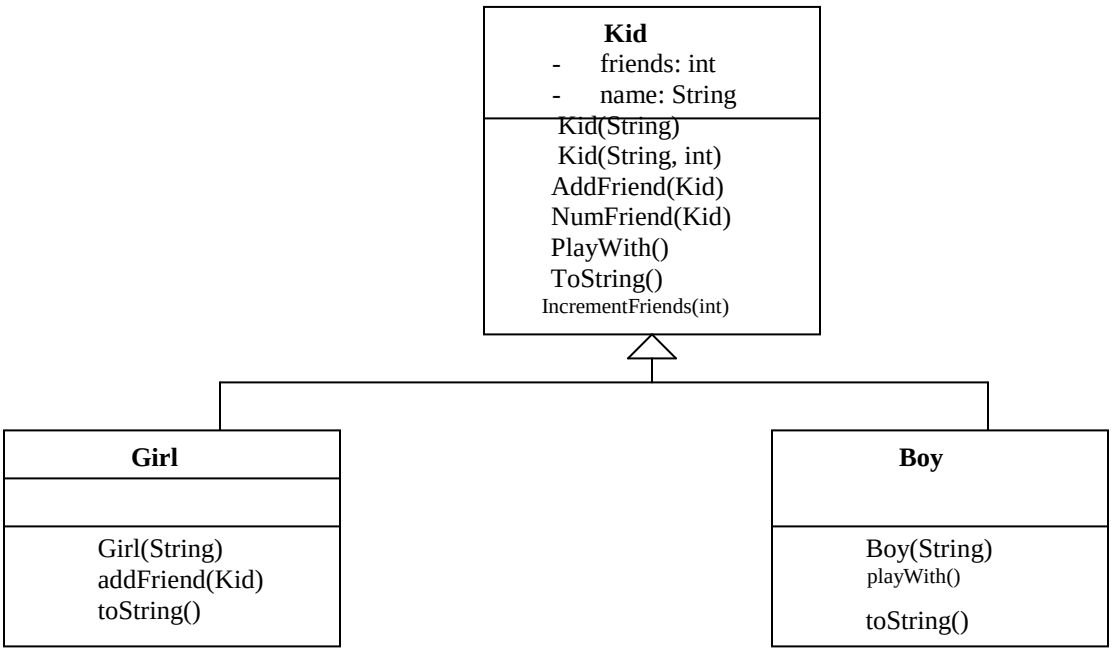


Q4. Consider the following class definitions and answer the questions that follow.

```
public class Kid {
   protected String name;
   private int friends;

   public Kid(String name) {
     this(name, 0);}

   public Kid(String n, int f) {
     this.name = n;
     friends = f;}

   public void addFriend(Kid k) {
     friends++;
   }
   public int numFriends() {
     return friends;
   }
   public String playsWith() {
     return "toys";
   }
   public String toString() {
     return "a kid named " + name;
   }
   public void incrementFriends(int n)
   { friends += n; }
}

public class Girl extends Kid {
   public Girl(String name) {
      super(name, 1);
   }
   public void addFriend(Kid k) {
      if (k instanceof Girl)
         incrementFriends(k.numFriends());
      else
         incrementFriends(1);
   }
   public String toString() {
      return "a girl named " + name;
   }
}

public class Boy extends Kid {
   public Boy(String name) {
      super(name);
   }
   public String playsWith() {
```

```
      return (super.playsWith() + " frogs");
   }
   public String toString() {
      return "a boy named " + name;
   }
}
```

Draw the UML class diagram for the three classes given above. (5 marks)

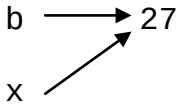


Q5. Given the following code and call sequence, give the output, assuming (10 marks)

a. Static scoping
a: 42 b: 84
b. Dynamic scoping
a: 4 b: 21

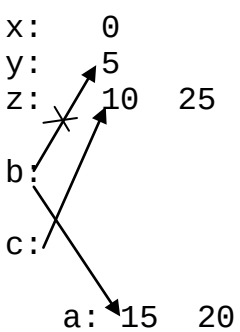
2.5 marks each

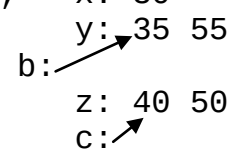
```
// Beginning of program file...
int a = 42, b = 84;

void Foo(int x) {
int a = 4, b = 27;
Bar(b);
}

void Bar(int& x) {
x = 21;
cout << “a: ” << a << “ ”;
cout << “b: ” << b << endl;
}

void main(){
Foo(a);
}
```



Q6. What does the following C++ program output? (10 marks)

```
#include <iostream>

using namespace std;

void f1(int, int*, int&);
int f2(int, int*, int&);

int x = 0;

int main() {
  int y = 5;
  int z = 10;

  f1(x, &y, z);

  cout << x << ", " << y << ", " << z << endl;

  x = 30;
  y = 35;
  z = 40;

  x = f2(x, &y, z);

  cout << x << ", " << y << ", " << z << endl;
}

void f1(int a, int *b, int &c) {
  a = 15;
  b = &a;
  *b = 20;
  c = 25;
}

int f2(int a, int *b, int &c) {
  a = 50;
  *b = 55;
  c = a;

  return a;
}
```





Answer:

0, 5, 25 **(1 mark, 1 mark and 2 mark)**
50, 55, 50 **(2 marks each)**

Q7. Complete the function body for “DoubleArray” below. Please note: you may not use the realloc function. (5 marks)

```
/* Doubles the size of int array oldArray, of length n. Returns
a pointer to a new array of length 2n, with the first n elements
copied from oldArray. Frees the memory held by
* oldArray.
* Requires: oldArray is an array of ints of length n.
*/

int *DoubleArray(int *oldArray, int n) {

/* Algorithm: create new array, copy old values over, free
* old array */

/* create new array */
int i=0, *newArray = (int*)malloc(sizeof(int)*2*n);
(1 marks)

/* copy old data */
for (i=0;i<n;i++) {newArray[i] = oldArray[i]; }
(2 mark)

/* note that it would be nice here to set elements n..2n-1
       to 0, but it’s not promised by the function contract */

free(oldArray); (1 mark)

return newArray; (1 mark)
}
```