

University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2012

Final Examination

Examiner: T.S. Abdelrahman, V. Betz, and M. Stumm

Duration: Two and a Half Hours

This exam is OPEN Textbook and CLOSED notes. The use of computing and/or communicating devices is NOT permitted.

Do not remove any sheets from this test book. Answer all questions in the space provided. No additional sheets are permitted.

Work independently. The value of each part of each question is indicated. The total value of all questions is 100.

Write your name and student number in the space below. Do the same on the top of each sheet of this exam book.

Name: _____
(Underline last name)

Student Number: _____

Q1. _____

Q7. _____

Q2. _____

Q8. _____

Q3. _____

Q9. _____

Q4. _____

Q10. _____

Q5. _____

Q11. _____

Q6. _____

Q12. _____

Total

Question 1. (15 marks). *General.*

Answer the following questions either by **circling** the correct answer. No justification of your answer is required.

(a) **True or False?** The elements of an array must all be of the same data type.

(b) **True or False?** The size of an array must be known at compile time (as opposed to at run time).

(c) **Circle one or more answers.** The variable `catname` is declared as `“string catname;”`. Which of the following is valid:

1. `catname = “Spanky”;`
2. `strcpy(catname, “Spanky”);`
3. `catname(“Spanky”);`
4. all of the above
5. none of the above

(d) **True or False?** C++ never provides a default constructor automatically.

(e) **Circle one or more answers.** Which of the following statements is correct:

1. a constructor is called at the time an object is declared
2. a constructor is called at the time an object is used
3. a constructor is called at the time a class is defined
4. a constructor is called at the time a class is used
5. none of the above

(f) **Circle one answer.** How many destructors may a class have?

1. zero.
2. one.
3. four.
4. any number as defined by the programmer.

(g) **True or False?** The copy constructor is always called when an object is passed by value to a function.

(h) **Circle one or more answers.** Members of a class specified as _____ are accessible only to methods of the class.

1. private.
2. public.
3. protected.
4. derived.
5. all of the above.

(i) **True or False?** A derived class inherits all the data members of its base class.

(j) **True or False?** A derived class inherits all the constructors of its base class.

(k) **True or False?** The following C++ statement is incorrect because the function must have an `int` return type, not `void`:

```
void virtual foo( int x, float y ) = 0;
```

(l) **True or False?** If you destroy an object through a pointer to a base class, and the base-class destructor is not virtual, the derived-class destructor is not executed.

(m) **True or False?** It is possible for a base class to also be derived from another class.

(n) **Circle one answer.** What is the best definition of a collision in a hash table?

1. Two entries are identical except for their keys.
2. Two entries with different data have the exact same key.
3. Two entries with different keys have the same exact hash value.
4. Two entries with the exact same key have different hash values.

(o) **Circle one answer.** You wish to store data in a hash table, where the data is identified by a string name. To do so, you require a function to compute an integer `hashCode` from the string name. The size of (number of entries in) the hash table is given by an integer `hashSize`. Which of the following hash functions below is best.

1. `int hashCode1 (string name) { return (name[0] * hashSize); }`
2. `int hashCode2 (string name) {return ((name[0] * 31)%hashSize);}`
3. `int hashCode3 (string name) { return (name[0]); }`
4. `int hashCode4 (string name) { int hval = 0; for (int i=0;i<name.length();i++) hval = hval * 13 + name[i]; return (hval % hashSize); }`

Question 2. (8 marks). *Objects with pointers.*

Study the following definitions and implementations of two classes Node and UseNode.

```
class Node {
private:
    int a;
    int b;
    Node* next;
public:
    void set_a(int x);
    void set_b(int y);
    void set_next(Node* n);
    Node* get_next();
    void print();
};

void Node::set_a(int x) {a = x;}
void Node::set_b(int y) {b = y;}
void Node::set_next(Node* n) {next = n;}
Node* Node::get_next() {return next;}
void Node::print() {cout << a << " " << b << " + "};}

class UseNode {
private:
    Node* head;
public:
    UseNode();
    UseNode(const UseNode & src);
    Node* getHead();
    void setHead(Node* h);
    void MoveHead();
    void print();
};

UseNode::UseNode() {head = NULL;}
UseNode::UseNode(const UseNode & src) {
    head = src.head->get_next();
}

Node* UseNode::getHead() {return head;}

void UseNode::setHead(Node* h) {head = h;}

void UseNode::MoveHead() {
    head = head->get_next()->get_next();
}

void UseNode::print() {
    Node* ptr = head;
    while (ptr!=NULL) {ptr->print(); ptr = ptr->get_next();}
    cout << endl;
}
```

Now consider the following main function, which use the above classes.

```
int main () {  
    Node* first = new Node;  
    first->set_a(5);  
    first->set_b(8);  
  
    Node* second = new Node;  
    second->set_a(9);  
    second->set_b(12);  
    first->set_next(second);  
  
    Node* third = new Node;  
    third->set_a(4);  
    third->set_b(10);  
    second->set_next(third);  
  
    Node* fourth = new Node;  
    fourth->set_a(3);  
    fourth->set_b(15);  
    fourth->set_next(NULL);  
    third->set_next(fourth);  
  
    UseNode MyNodes;  
    MyNodes.setHead(first);  
  
    UseNode HisNodes = MyNodes;  
    UseNode HerNodes(HisNodes);  
    UseNode TheirNodes;  
  
    TheirNodes = HisNodes;  
    TheirNodes.MoveHead();  
  
    MyNodes.print(); // Statement # 1  
  
    HisNodes.print();// Statement # 2  
  
    HerNodes.print();// Statement # 3  
  
    TheirNodes.print();// Statement # 4  
  
    return (0);  
}
```

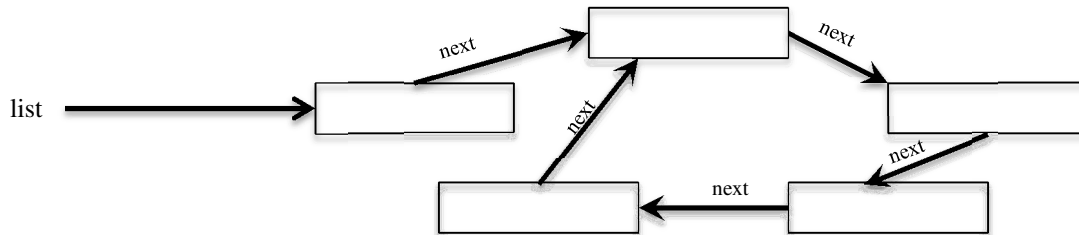
Output

Indicate the output printed by each numbered statement in the above main function. Write the output in the table shown above next to the code.

Hint: Draw a picture.

Question 3. (8 marks). *Linked Lists.*

Write a function, `bool ListNode::cycle()`, that detects whether a linked list has a *cycle*. If the next field of a node is set to the address of one of the “previous” nodes on the list, a cycle exists, as shown in the figure below.



where `ListNode` has a definition as follows:

```
class ListNode {
private:
    int value ;
    ListNode* next;

public:
    ListNode() { value = 0 ; next = NULL ; }
    ListNode( int v ) { value = v ; next = NULL ; }
    ListNode( int v, ListNode *n ) { value = v ; next = n ; }
    ~ListNode() ;
    bool cycle() ; // returns true if cycle is detected; false otherwise
    .
    .
    .
};
```

Your solution may require additional private data members for `ListNode`. If so, list them and indicate how they should be initialized in the constructors. Otherwise, write NONE.

Declare the additional private members you wish to add to `ListNode` here:

Show how these members should be initialized in the constructors:

Write the body of the cycle function below.

```
bool ListNode::cycle() {
```

```
}
```

Question 4. (8 marks). *Trees.*

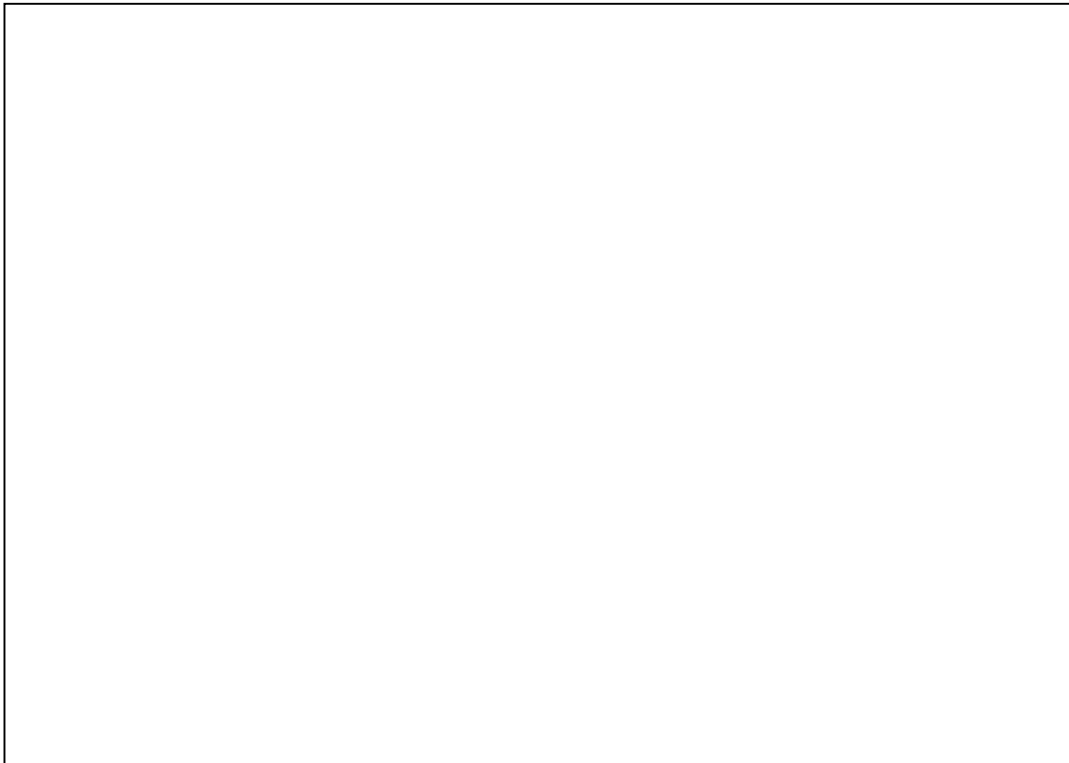
Consider the following definition of the class `treenode`.

```
class treenode {
public:
    int data;
    treenode *left;
    treenode *right;
    int count_leaves();
};

treenode *Root; // root of the tree
```

Write a **recursive member** function `count_leaves` that returns the number of leaf nodes in the tree.

```
int treenode::count_leaves () {
```



```
}
```

```
// This is how count_leaves is called
cout << Root->count_leaves();
```


Question 5. (8 marks). *Recursion.*

The `Screen` class below represents the pixels on a screen as a two-dimensional array of integers; the color of the pixel at location (x, y) is represented by `pixel[x][y]`. A common graphics operation is a *flood-fill*, where all the pixels forming a connected shape of a certain initial color are changed to another (final) color. You are to implement the recursive flood function declared in the class.

```
#define Ndim 7

class Screen {
private:
    int pixel[Ndim][Ndim];

public:
    void flood (int x, int y, int initColor, int finalColor);
    void setPixel (int x, int y, int color);
    int getPixel (int x, int y);
};

void Screen::setPixel (int x, int y, int color) {pixel[x][y]=color;}
int Screen::getPixel (int x, int y) {return pixel[x][y];}
```

Note that two pixels must be either vertically or horizontally adjacent and the same color to be considered part of the same shape; the flood algorithm will not consider diagonally adjacent pixels to be connected. Below is an example, showing how an initial state of the pixel array would be changed by the flood call shown in main.

```
int main () {
    Screen s;
    // Set the value of each pixel

    s.flood (0, 5, 1, 3);
}
```

pixel: before call to flood() in main()

0	0	0	0	0	0	0	y = 6
1	0	0	2	2	0	0	
1	1	2	1	1	0	0	
0	1	0	1	1	0	0	
1	0	0	0	1	0	0	
0	0	0	0	1	1	0	
0	0	0	0	1	1	1	y = 0
x = 0						x = 6	

pixel: after call to flood() in main()

0	0	0	0	0	0	0	y = 6
3	0	0	2	2	0	0	
3	3	2	1	1	0	0	
0	3	0	1	1	0	0	
1	0	0	0	1	0	0	
0	0	0	0	1	1	0	
0	0	0	0	1	1	1	y = 0
x = 0							

Note that if the pixel at location (x, y) is not of the initial color `initColor`, then function returns without changing any pixels. For example, the call `s.flood(0, 0, 1, 3);` does not change the pixel array.

Write the flood member function. Clearly show both the function header and the function body.

Question 6. (10 marks). *Tree Traversals.*

- (a) **(3 marks).** Draw the binary search tree that results from inserting nodes with the following values in the order they are given (i.e., from left to right):

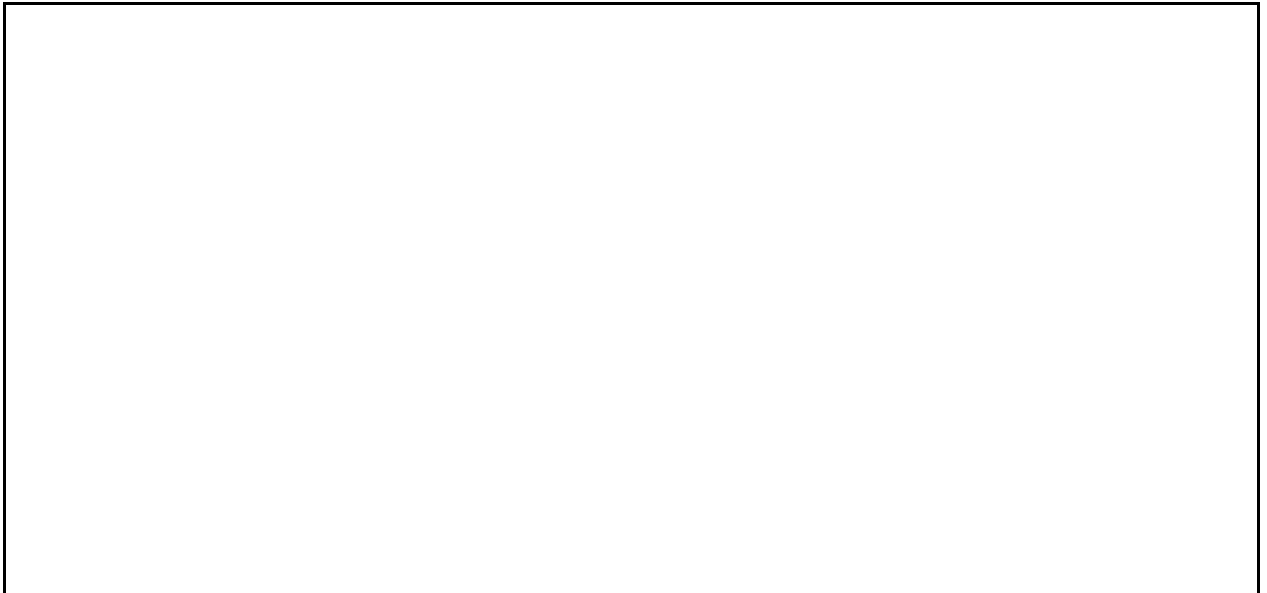
6 7 8 9 10 5 3 4 2 1

- (b) **(3 marks).** Then in the tree obtained in (a) above, delete the node containing the value 3 and reinsert a new node with the value 3 again. Draw the resulting tree:

- (c) **(4 marks)**. For a tree whereby leaves only contain integers as values and interior nodes only contain operator symbols (i.e., +, -, *, /, %) as values, a post-order traversal produced the following output:

2 5 + 3 7 + * 5 7 * -

Draw the tree:



Question 7. (10 marks). *Inheritance.*

Consider the following definitions and implementations of three classes `Animal`, `Mammal` and `Carnivore`.

```
// Class Animal
class Animal {
    private:
        string name;

    public:
        Animal(string n);
        virtual ~Animal();
        void print();
};

Animal::Animal(string n) {
    name = n;
    cout << name << " is an animal" << endl;
}

Animal::~~Animal() {
    cout << "The " << name << " animal is gone" << endl;
}

void Animal::print() {
    cout << name << " is an animal" << endl;
}

// Class Mammal
class Mammal : public Animal {
    private:
        string m_name;

    public:
        Mammal(string n);
        virtual ~Mammal();
        void print();
};

Mammal::Mammal(string n) : Animal (n) {
    m_name = n;
    cout << m_name << " is a mammal" << endl;
}

Mammal::~~Mammal() {
    cout << "The " << m_name << " mammal is gone" << endl;
}

void Mammal::print() {
    cout << m_name << " is a mammal" << endl;
}
```

```

// Class Carnivore
class Carnivore : public Mammal {
    private:
        string c_name;

    public:
        Carnivore (string n);
        virtual ~Carnivore ();
        void print();
};

Carnivore::Carnivore (string n) : Mammal (n) {
    c_name = n;
    cout << c_name << " is a Carnivore " << endl;
}

Carnivore::~~Carnivore () {
    cout << "The " << c_name << " carnivore is gone" << endl;
}

void Carnivore::print() {
    cout << c_name << " is a Carnivore " << endl;
}

```

Consider the following code in main. The code is shown in a table to facilitate writing your answers. Indicate the output generated when each line of the code is executed. If no output is generated, write **NONE**. There are no compile-time errors in the code.

Code	Output
using namespace std;	NONE
#include <iostream>	NONE
#include <string>	NONE
int main() {	NONE
bool isTrue = true;	
if (isTrue) {	
string cow = "Cow";	
Animal Cow(cow);	
string camel = "camel";	
Mammal Camel(camel);	
string tiger = "Tiger";	

<code>Carnivore Tiger(tiger);</code>	
<code>}</code>	
<code>Animal* animal_ptr;</code>	
<code>Mammal* mammal_ptr;</code>	
<code>string cat = "Cat";</code>	
<code>mammal_ptr = new Mammal(cat);</code>	
<code>animal_ptr = mammal_ptr;</code>	
<code>animal_ptr->print();</code>	
<code>delete animal_ptr;</code>	
<code>return(0);</code>	
<code>}</code>	

Question 8. (10 marks). *Inheritance and Polymorphism.*

What is the output generated by the following code:

```
class A {
private:
    int a ;
protected:
    A() { cout << "A ctor" << endl ; }
public:
    A(int i) { a=i ; cout << "A(int) ctor" << endl ; }
    virtual ~A() { cout << "A dtor" << endl ; }
} ;

class B: public A {
private:
    int b ;
public:
    B() { cout << "B ctor" << endl ; }
    B( int i ) { b=i ; cout <<
                "B(int) ctor" << endl;}
    B( int i, int j):A(i) { b=j; cout <<
        "B(int,int) ctor" << endl ; }
    ~B() { cout << "B dtor" << endl ; }
    void virtual dummy(){ A a(3) ; }
} ;

class C: public B{
private:
    int c ;
public:
    C() { cout << "C ctor" << endl ; }
    void dummy() { A *ap = new A(2) ; }
} ;

main() {
    A *aap1 = new A(2) ;    cout << "-" << endl;
    B *bbp0 = new B ;      cout << "-" << endl;
    B *bbp2 = new B(2,3);  cout << "-" << endl;
    B *bcp0 = new C ;      cout << "-" << endl;
    C *ccp0 = new C ;      cout << "-" << endl;
    bcp0->dummy() ;        cout << "-" << endl;
    bbp2->dummy() ;        cout << "-" << endl;
    delete aap1 ;          cout << "-" << endl;
    delete bbp2 ;          cout << "-" << endl;
    delete ccp0 ;          cout << "-" << endl;
}
```

Output:

Question 9. (11 marks). *Complexity Analysis.*

- (a) **(3 marks).** Write the execution time, $T(n)$ where n is the problem size, of the code below in big O notation.

```
int val = 0;
for (int i = 0; i < n; i++) {
    int j = 0;
    while (j * j < n) {
        for (int k = 0; k < n/2; k++) val -= k;
        j = j + 1;
    } // End while j loop
} // End for i loop
```

$T(n) = O(\quad)$

- (b) **(3 marks).** Consider the following recursive function, where n is the problem size.

```
int odd_divide(int n){
    if( n==1 ) return 1;
    return (odd_divide(n/2) + 2*odd_divide(n/2) );
}
```

Write the recurrence equation for the execution time of the above function, $T(n)$.

$T(n) =$

What is the execution time $T(n)$ expressed using big- O notation?

$T(n) = O(\quad)$

(c) (3 marks). Consider the following recursive function, where n is the problem size:

```
int foo(int n){
    if( n==0 ) return 1;
    int sum = 0;
    for(int i = 0; i < n; ++i) sum += foo(i);
    return sum;
}
```

Write the recurrence equation for the execution time of the above function, $T(n)$.

$T(n) =$

What is the execution time $T(n)$ expressed using big-O notation?

$T(n) = O(\quad)$

(d) (2 marks). A database designer must support an equal mix of insert, search (i.e., find) and delete (remove) operations on a set of n elements, where n is very large. Which of the following data organizations is best from a compute time perspective. Circle one answer:

Unsorted array

Sorted array

Balanced (i.e., perfect or minimum level) tree

Unsorted linked list

Sorted linked list

Question 10. (3 marks). *Hash Tables.*

Assume a hash table with size $T = 11$ with the following hash function:

$$h: \text{index} = (\text{key} \bmod T) \quad (\text{note: mod is the modulus operator, \% in C++})$$

Show the contents of the hash table after the following operations have been performed. Indicate next to each operation the number of ***probes*** performed. Assume that linear probing is used to resolve collisions and that the hash table is initially empty.

1. insert 14
2. insert 26
3. insert 4
4. delete 26
5. search 4
6. insert 37

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Question 11. (3 marks). *Hash Tables.*

A hash table with size $T = 7$ has the following hash function:

$$h: \text{index} = (\text{key} \bmod 7) \quad (\text{note: mod is the modulus operator, \% in C++})$$

The hash table is shown below (left side) after some elements have been hashed into the table. Also some elements have been deleted; they are indicated with a “—” across the element.

0	21
1	8
2	9
3	10
4	3
5	12
6	18

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

It is desired to expand the table to be of size $T = 11$ and re-insert the elements on it. The new table has the following hash function:

$$h: \text{index} = (\text{key} \bmod 11) \quad (\text{again, mod is the modulus operator, \% in C++})$$

Show the contents of the new table after the elements have been re-inserted. Write your answers in the empty hash table above (right side).

Question 12. (6 marks). *Debugging memory problems.*

Consider the code below, which is in a file called `memoryFaults.cpp` and which also has line numbers displayed. The code declares a class `A`. This class is intended to be an array that knows how many data items (`N`) it contains. It is also intended to have a proper deep copy assignment operator, destructor and a `clear()` function to make the array have no valid data.

The engineer who wrote this code suspects it contains bugs in how it uses memory, and has run `valgrind` on it. The `valgrind` output is shown after the code and indicates several problems.

```
1      #include <iostream>
2      using namespace std;
3
4      class A {
5      private:
6          int *array;
7          int N;
8      public:
9          A(int _N);
10         ~A();
11         A & operator= (const A& rhs);
12         int& operator[] (int index);
13         void clear ();
14         int getN () const;
15         void print () const;
16     };
17
18     A::A (int _N) {
19         N = _N;
20         array = new int[N];
21     }
22
23     A::~~A () {
24         delete array;
25     }
26
27     A& A::operator= (const A& rhs) {
28         for (int i = 0; i < rhs.N; i++)
29             array[i] = rhs.array[i];
30
31         N = rhs.N;
32         return (*this);
33     }
34
35     int& A::operator[] (int index) {
36         return (array[index]);
37     }
38
39     void A::clear () {
40         delete[] array;
41         N = 0;
42     }
43
44     void A::print () const {
45         for (int i = 0; i <= getN(); i++)
46             cout << array[i];
47         cout << endl;
```

```

48     }
49
50     int A::getN () const {
51         return N;
52     }
53
54     A setup () {
55         A* ap = new A(4);
56         return (*ap);
57     }
58
59     int main () {
60         cout << "Main: initialize a1\n";
61         A a1(5);
62         for (int i = 0; i < 5; i++)
63             a1[i] = i;
64         cout << "Main: initialize a2\n";
65         A a2(4);
66         for (int i = 0; i < 4; i++)
67             a2[i] = i + 1;
68         cout << "Main: update and print a2\n";
69         a2 = a1;
70         a2 = setup ();
71         a2.print();
72         cout << "Main: start cleanup\n";
73         a1.clear();
74         cout << "Ending main\n";
75         return 0;
76     }

```

Valgrind output when run on this program:

```

Main: initialize a1
Main: initialize a2
Main: update and print a2
==3391== Invalid write of size 4
==3391==   at 0x8048930: A::operator=(A const&) (memoryFaults.cpp:29)
==3391==   by 0x8048B46: main (memoryFaults.cpp:69)
==3391== Address 0x42d3080 is 0 bytes after a block of size 16 alloc'd
==3391==   at 0x402532E: operator new[](unsigned int) (vg_replace_malloc.c:299)
==3391==   by 0x80488CD: A::A(int) (memoryFaults.cpp:20)
==3391==   by 0x8048AEE: main (memoryFaults.cpp:65)
==3391==
==3391== Mismatched free() / delete / delete []
==3391==   at 0x4024851: operator delete(void*) (vg_replace_malloc.c:387)
==3391==   by 0x8048900: A::~~A() (memoryFaults.cpp:24)
==3391==   by 0x8048B71: main (memoryFaults.cpp:70)
==3391== Address 0x42d30e8 is 0 bytes inside a block of size 16 alloc'd
==3391==   at 0x402532E: operator new[](unsigned int) (vg_replace_malloc.c:299)
==3391==   by 0x80488CD: A::A(int) (memoryFaults.cpp:20)
==3391==   by 0x8048A26: setup() (memoryFaults.cpp:55)
==3391==   by 0x8048B51: main (memoryFaults.cpp:70)
==3391==
==3391== Invalid read of size 4

```

```
==3391== at 0x80489AE: A::print() const (memoryFaults.cpp:46)
==3391== by 0x8048B7C: main (memoryFaults.cpp:71)
==3391== Address 0x42d3080 is 0 bytes after a block of size 16 alloc'd
==3391== at 0x402532E: operator new[](unsigned int) (vg_replace_malloc.c:299)
==3391== by 0x80488CD: A::A(int) (memoryFaults.cpp:20)
==3391== by 0x8048AEE: main (memoryFaults.cpp:65)
==3391==
00004
Main: start cleanup
Ending main
==3391== Mismatched free() / delete / delete []
==3391== at 0x4024851: operator delete(void*) (vg_replace_malloc.c:387)
==3391== by 0x8048900: A::~~A() (memoryFaults.cpp:24)
==3391== by 0x8048BBF: main (memoryFaults.cpp:75)
==3391== Address 0x42d3070 is 0 bytes inside a block of size 16 alloc'd
==3391== at 0x402532E: operator new[](unsigned int) (vg_replace_malloc.c:299)
==3391== by 0x80488CD: A::A(int) (memoryFaults.cpp:20)
==3391== by 0x8048AEE: main (memoryFaults.cpp:65)
==3391==
==3391== Invalid free() / delete / delete[]
==3391== at 0x4024851: operator delete(void*) (vg_replace_malloc.c:387)
==3391== by 0x8048900: A::~~A() (memoryFaults.cpp:24)
==3391== by 0x8048BCA: main (memoryFaults.cpp:75)
==3391== Address 0x42d3028 is 0 bytes inside a block of size 20 free'd
==3391== at 0x40244D3: operator delete[](void*) (vg_replace_malloc.c:409)
==3391== by 0x8048985: A::clear() (memoryFaults.cpp:40)
==3391== by 0x8048B9B: main (memoryFaults.cpp:73)
==3391==
==3391==
==3391== HEAP SUMMARY:
==3391== in use at exit: 8 bytes in 1 blocks
==3391==
==3391== 8 bytes in 1 blocks are definitely lost in loss record 1 of 1
==3391== at 0x402569A: operator new(unsigned int) (vg_replace_malloc.c:255)
==3391== by 0x8048A12: setup() (memoryFaults.cpp:55)
==3391== by 0x8048B51: main (memoryFaults.cpp:70)
==3391==
==3391== LEAK SUMMARY:
==3391== definitely lost: 8 bytes in 1 blocks
```

In the table below, indicate the changes required to the code to fix the bugs. **No changes should be made to the main() function; all fixes can be made with changes to the other functions. Also, the changes should not alter the intent of the class A to represent an array.**

Line Number	Remove / Replace with / Insert Before (choose one)	Code to Insert or Replace with (Can be More than One Line)