CS 31 Worksheet 6 Solutions

This worksheet is entirely optional, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. All exams will be done on paper, so it is in your best interest to practice these problems by hand and not rely on a compiler.

If you have any questions or concerns please email arabellekezia@ucla.edu or luluw@ucla.edu, or go to any of the LA office hours.

Concepts

Structs/Class, Public/Private, Simple Constructors, Dynamic Allocation

1. Conceptual Questions
   - What's the main difference between declaring a type with the keyword `struct` and declaring it with the keyword `class`? struct has public variables (unless declared otherwise) and class has private variables (unless declared otherwise)
   - Why should you not allow data members to be public? to hide their implementation from other classes or structs and ensure that no other classes or structs leave data members in an invalid state/configuration
   - What is the purpose of having private member functions in a class? Can you give some examples of when they would be used? used in functions that are only used within the class
   - What happens if you forget to deallocate memory once you're done with the object? memory leaks
   - (True/False) A class may have more than one constructor. True
   - (True/False) A class may have more than one destructor. False
   - If you have an object pointed by a pointer, which operator is used with the pointer to access the object's members? ->

Time: 3 mins

2. Write a class Person that has two private data members:
   - `m_age` (an int)
   - `m_catchphrase` (a string).

   The Person class should have a default constructor that initializes its data members to reasonable values and a second constructor that initializes the data members to the values of its parameters. In addition, Person should have three public member functions:
   - `getAge()`, which returns the Person's age
   - `haveBirthday()`, which increments the Person's age by 1
   - `speak()`, which prints the Person's catchphrase.

Time: 4 mins

```
class Person{
        private:
                int m_age;
                string m_catchphrase;
        public:
                Person() {
                        m_age = 0;
                        m_catchphrase = "";
                }
                Person (int age, string catchphrase) { // need this for problem 4
                        m_age = age;
                        m_catchphrase = catchphrase;
                }
                int getAge() {
                        return m_age;
                }
                void haveBirthday() {
                        m_age++;
                }
                void speak() {
                        cout << m_catchphrase;
                }
}
```

3. A line in Euclidean space can be represented by two parameters, **m** and **x** from its slope-intercept equation **y = mx + b.** Here **m** represents the slope of the line and **b** represents the line's y-intercept.
   Write a class that represents a line. Your class must have a simple constructor that initializes the line's **m** and **b**. Next, define a member function with the following prototype:
   `double intersection(Line line2);`

   This function must compute the x-coordinate where this line and another line (line2) intersect.

   ```
   double m1 = 2;
   double b1 = 3;
   double m2 = -2;
   double b2 = 7;
   Line line1(m1, b1);
   Line line2(m2, b2);
   cout << line1.intersection(line2) << endl;    // prints 1.0
   ```

   This function must compute the x-coordinate where this line and another line (line2) intersect.

Time: 6 mins

```
class Line
{
public:
        Line(double m, double b)
        {
                m_m = m;
                m_b = b;
        }

        double m() const
        {
                return m_m;
        }

        double b() const
        {
                return m_b;
        }

        double intersection(Line line2)
        {
                if (m_m == line2.m())
                {
                        // same slope! SO the lines either are coincident or parallel
                        // spec doesn't specify what we should do here, so return
                        // whatever; in the real world we may want to throw an
                        // exception, (which aren't discussed in CS 31)
                        return 0;
                }
                return (line2.b() - m_b)/(m_m - line2.m());
        }

private:
        double m_m;
        double m_b;
};
```

Bonus: There are two or three ways in which this problem specification is incomplete; they are not related to C++, but to the problem domain. What are they?

As mentioned in the comments above, the spec does not tell us what we should return if the two lines are coincident or parallel. Also, vertical lines cannot be exactly defined using the framework we have (e.g. x=3), although they can be approximated using a line with a large m.

4. Write a program that repeatedly reads an age and a catchphrase from the user and uses them to dynamically allocate a Person object, before calling the Person's speak() function and then deallocating the Person object.

   Time: 4 mins

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

int main()
{
      int age;
      string catchphrase;
      while(true)
      {
            cout << "Please enter an age: ";
            cin >> age;
            cin.ignore(10000, '\n');
            cout << "Please enter a catchphrase: " << endl;
            getline(cin, catchphrase);
            Person* p = new Person(age, catchphrase);
            p->speak();
            delete p;
      }
}
```

5. Find the **six** errors in the following code, and write the fixes.

```
const int NAME_LEN = 100;

class Cat {
      int m_age;
      char m_name[NAME_LEN];
      string m_type;
public: //(1)
      Cat(int age, const char name[], string type) {
            m_age = age;
            m_name = name;          //(2)
            strcpy(m_name, name); //(2)
            m_type = type;
```

```
	}

	void introduce() {
		cout << "Hi! I am a " + m_type + " cat" << endl;
	}
};


struct Sheep {
	string m_name;
	int m_age;
	Sheep(int age) {
		m_age = age;
	}
	void introduce() {
		cout << "Hi! I am " + m_name + " the sheep" << endl;
	}
} (3)
}; // Don't forget the semicolon!(3)

int main() {
	Cat* schrodinger = new Cat(5, "Schrodinger's cat", "Korat");
	schrodinger->introduce();
	cout << schrodinger->m_age << endl; // Private variables cannot
	                                    // be accessed outside a
	                                    // class declaration. (4)
	Sheep dolly(6);
	dolly->introduce(); //(5)
	dolly.introduce();  //(5)

	delete schrodinger;
	delete dolly; (6)
	 // Do not delete dolly, because it was not created on the heap!
	// Every execution of delete should correspond to an execution
	// of new. (6)
}
```

What will the program above successfully print once all the fixes have been made?

Time: 5 mins

Hi! I am a Korat cat!
Hi! I am the sheep!

6. Write a class called Complex, which represents a complex number. Complex should have a default constructor and the following constructor:

```
Complex(int real, int imaginary);
// -3 + 8i would be represented as Complex(-3, 8)
```

Additionally, the class should contain two functions: sum and print. Sum should add two complex numbers. Print should print which complex number the object represents. You may declare any private or public member variables or getters/setters you deem necessary. Your code should work with the example below.

```
int main() {
(1)   Complex c1(5, 6);
(2)   Complex c2(-2, 4);
(3)   Complex* c3 = new Complex();

(4)   c1.print();
(5)   c2.print();
(6)   cout << "The sum of the two complex numbers is:" << endl;
(7)   c3->sum(c1, c2);
(8)   c3->print();
(9)   delete c3;
}

// The output of the main program:
5+6i
-2+4i
The sum of the two complex numbers is:
3+10i
```

Time: 5 mins

```
class Complex {
    int m_real;
    int m_imaginary;
  public:
    Complex() {}
    Complex(int real, int imaginary) {
        m_real = real;
        m_imaginary = imaginary;
    }
    void print() {
        cout << m_real << "+" << m_imaginary << "i" << endl;
    }
    void sum(Complex c1, Complex c2) {
        m_real = c1.m_real + c2.m_real;
        m_imaginary = c1.m_imaginary + c2.m_imaginary;
    }
};
```

What would happen if swapped the order of (8) and (9)? How would it change the output?

Time: 2 mins

```
After deleting the object pointed to by c3, an attempt to follow
the pointer c3 is undefined behavior.  The program might crash,
print weird values (perhaps because the memory used by the
deleted object was overwritten with some bookkeeping information
the storage manager uses), print 3+10i (if the memory used was
not overwritten), or do something else.
```

7. *Write a class *TicTacToe* that simulates a game of Tic Tac Toe on a 3x3 board. (You may know the game as Noughts and Crosses, Xs and Os, Tres en Raya, OXO, 井字棋, 삼목, cờ ca rô, Крестики-нолики.) Your version of the class may implement any private members you wish it to, but the class should implement the following public members:
   - A default constructor that starts a game with an empty board with it being X's turn to play.
   - A constructor that takes in one parameter: `bool Xstarts`
     - If the parameter is true, the game starts with X's turn.
     - If the parameter is false, the game starts with O's turn.
     - Regardless of the parameter's value, the game should start with an empty board.
   - `bool placePiece(int r, int c)`
     - Places a game piece belonging to the current player (X or O) on the board at position r,c.
     - The top left of the board is position (1, 1) and the bottom right of the board is position (3, 3).
     - Returns true if successful and false if the piece cannot be placed.
     - A piece cannot be placed if another piece has already been placed in the position or if the current game has ended.
     - If this function returns true, the next turn will be the player other than the current player.
     - A game ends if three pieces of the same type exist on the same row, column, or diagonal, or if no board position is empty.
   - `bool isXTurn()`
     - Returns true if current turn belongs to X and false otherwise
   - `bool isGameEnded()`
     - Returns true if a player has won the game or if no board position is empty, and false otherwise.

- `char getWinner()`
  - Returns 'X' if X has won the current game, 'O' if O has won the game, '=' if neither has won and no board postion is empty.
  - Returns '?' if the current game has not ended.
- `void clear(bool Xstarts)`
  - Clears all game pieces from the board and starts a new game
  - If the parameter is true, the game starts with X's turn.
  - If the parameter is false, the game starts with O's turn.

Time: 12 mins

```
class TicTacToe {
public:
        TicTacToe();
        TicTacToe(bool Xstarts);
        bool placePiece(int r, int c);
        bool isXTurn();
        bool isGameEnded();
        char getWinner();
        void clear(bool Xstarts);

private:
        bool Xturn;
        int piecesPlaced;
        char winner;
        char board[3][3];
        const int BOARD_SIZE = 3;

        void initBoard();
        void updateWinner(int r, int c);
        char getCurrPiece();
};

TicTacToe::TicTacToe() {
        clear(true);
}

TicTacToe::TicTacToe(bool Xstarts) {
        clear(Xstarts);
}

bool TicTacToe::placePiece(int r, int c) {
        if (winner != '?' || board[r - 1][c - 1] != '-') {
                return false;
```

```
		}
		else {
			board[r - 1][c - 1] = getCurrPiece();
			piecesPlaced++;
			Xturn = !Xturn;
			updateWinner(r, c);
			return true;
		}
}

bool TicTacToe::isXTurn() {
		return Xturn;
}

bool TicTacToe::isGameEnded() {
		return winner != '?';
}

char TicTacToe::getWinner() {
		return winner;
}

void TicTacToe::clear(bool Xstarts) {
		Xturn = Xstarts;
		winner = '?';
		piecesPlaced = 0;
		initBoard();
}

void TicTacToe::initBoard() {
		for (int r = 0; r < BOARD_SIZE; r++) {
			for (int c = 0; c < BOARD_SIZE; c++) {
				board[r][c] = '-';
			}
		}
}


void TicTacToe::updateWinner(int r, int c) {
		char currPiece = board[r - 1][c - 1];
		bool rowWin = true;
		bool colWin = true;
		bool diagWin = true;
```

```
        for (int pos = 0; pos < BOARD_SIZE; pos++) {
                rowWin = rowWin && (currPiece == board[pos][c - 1]);
                colWin = colWin && (currPiece == board[r - 1][pos]);
        }

        if (r == c) {
                for (int pos = 0; pos < BOARD_SIZE; pos++) {
                        diagWin = diagWin && (currPiece == board[pos][pos]);
                }
        }
        else if (r + c == BOARD_SIZE + 1) {
                for (int pos = 0; pos < BOARD_SIZE; pos++) {
                        diagWin = diagWin && (currPiece == board[BOARD_SIZE -
                                                                        1 - pos][pos]);
                }
        }
        else {
                diagWin = false;
        }

        if (rowWin || colWin || diagWin) {
                winner = currPiece;
        }
        else if (piecesPlaced == 9) {
                winner = '=';
        }
}

char TicTacToe::getCurrPiece() {
        if (Xturn)
                return 'X';
        else
        return 'O';
}
```

8. Suppose you have a struct defined as follows:

```
struct Array {
        int* vals;
        int len;
}
```

Within Array, `vals` is a pointer to an array of ints (that is *not* dynamically allocated). The field `len` describes the length of this array.

Design a function with the following header:

```
int findArrayWithMax(Array arr1, Array arr2, Array arr3);
```

Given three Arrays *arr1*, *arr2*, and *arr3*, this function should return the number of the Array that contains the maximum value of the three Arrays. If the Array with the maximum value is *arr1*, it should return 1 (2 for *arr2* and 3 for *arr3*). *Hint: You may write a helper function*

```
int a[5] = {3, 4, 5, 6, 1};
int b[2] = {1000, -1};
int c[9] = {23, 2, 1, 4, 65, 42, 10, -20, 7};
Array arr1 = { a, 5 };
Array arr2 = { b, 2 };
Array arr3 = { c, 9 };
int max = findArrayWithMax(arr1, arr2, arr3); // max = 2
```

```
Time: 6 mins
```

```
void updateMax(Array arr, int& max, int& maxNum, bool& numFound, int arrNum)
{
        for (int x = 0; x < arr.len; x++) {
                if (!numFound || arr.vals[x] > maxNum) {
                        max = arrNum;
                        maxNum = arr.vals[x];
                        numFound = true;
                }
        }
}

int findArrayWithMax(Array arr1, Array arr2, Array arr3) {
        int max = 0;
        int maxNum;
        bool numFound = false;

        updateMax(arr1, max, maxNum, numFound, 1);
        updateMax(arr2, max, maxNum, numFound, 2);
        updateMax(arr3, max, maxNum, numFound, 3);
        return max;
}
```

*Challenge problem