```objc
/**
 *  main.m
 *  Section0App
 *  Tommy MacWilliam, 2011
 *
 *  Create several students and assign them to a TF, then have the TF grade them and schedule office hours accordingly
 *
 */

#import <Foundation/Foundation.h>
#import "TF.h"
#import "Student.h"

int main (int argc, const char * argv[])
{

    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    // create student objects
    Student *alycia = [[Student alloc] initWithName:@"Alycia"];
    Student *joe = [[Student alloc] initWithName:@"Joe"];

    // create array of students
    NSMutableArray *students = [[NSMutableArray alloc] initWithObjects: alycia, joe, nil];

    // create new TF
    TF *tommy = [[TF alloc] initWithName:@"Tommy" students:students favorite:alycia];
    NSLog(@"%@", tommy.favorite.name);

    // give TF another student
    Student *tim = [[Student alloc] initWithName:@"Tim"];
    [students addObject:tim];

    // assign and output grades
    [tommy grade];
    [tommy outputGrades];

    // create and output office hours
    [tommy addOfficeHour:[NSDate dateWithNaturalLanguageString:@"1/23/45"]];
    [tommy addOfficeHour:[NSDate dateWithNaturalLanguageString:@"4/56/78"]];
    [tommy outputOfficeHours];

    // release everything
    [alycia release];
    [joe release];
    [students release];
    [tommy release];
    [tim release];

```

```
49.     [pool drain];
50.     return 0;
51. }
52.
53.
```

```m
/**
 *  Student.m
 *  Section0App
 *  Tommy MacWilliam, 2011
 *
 */

#import "Student.h"


@implementation Student

@synthesize name = _name;

/**
 * Initialize a new student
 *
 */
- (id)initWithName:(NSString *)newName {
    self = [super init];

    if (self != nil)
        self.name = newName;

    return self;
}

/**
 * Deallocate
 *
 */
- (void)dealloc {
    // release all properties
    self.name = nil;

    [super dealloc];
}

@end

```

```m
/**
 *  TF.m
 *  Section0App
 *  Tommy MacWilliam, 2011
 *
 */

#import "TF.h"
#import "Student.h"
#import <stdlib.h>

@implementation TF

@synthesize name = _name, students = _students, favorite = _favorite, grades = _grades, ohs = _ohs;

/**
 * Initialize a new TF
 *
 */
- (id)initWithName:(NSString *)newName students:(NSMutableArray *)newStudents favorite:(Student *)newFavorite {
    self = [super init];

    // make sure we have a valid object
    if (self != nil) {
        // set all properties
        self.name = newName;
        self.students = newStudents;
        self.favorite = newFavorite;

        // create a blank dictionary for grades
        self.grades = [[NSMutableDictionary alloc] init];
        self.ohs = [[NSMutableArray alloc] init];
    }

    return self;
}

/**
 * Assign grades to students
 *
 */
- (void)grade {
    // seed PRNG
    srand(time(NULL));

    // iterate through all students
    for (Student *student in self.students) {
        // after much thought and consideration, decide on a student's grade
```

```
49.         int grade = rand() % 4;
50.
51.         // favorite student gets a check+
52.         if ([student.name isEqualToString:self.favorite.name])
53.             grade = 3;
54.
55.         // add student/grade pair to dictionary
56.         [self.grades setValue:[NSNumber numberWithInt:grade] forKey:student.name];
57.     }
58. }
59.
60. /**
61.  * Display grades for all students
62.  *
63.  */
64. - (void)outputGrades {
65.     NSLog(@"Grades:");
66.
67.     // iterate over grades and output student/grade pair
68.     for (id student in self.grades)
69.         NSLog(@"%@: %@", student, [self.grades objectForKey:student]);
70. }
71.
72. /**
73.  * Add an office hour
74.  *
75.  */
76. - (void)addOfficeHour:(NSDate *)date {
77.     [self.ohs addObject:date];
78. }
79.
80. /**
81.  * Display all office hours
82.  *
83.  */
84. - (void)outputOfficeHours {
85.     NSLog(@"Office Hours:");
86.
87.     // create new date formatter to output NSDate in human-readable format
88.     NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
89.     [formatter setDateStyle:NSDateFormatterLongStyle];
90.
91.     // iterate over office hours and display each
92.     for (NSDate *oh in self.ohs)
93.         NSLog(@"%@", [formatter stringFromDate:oh]);
94.
95.     [formatter release];
96. }
```

```
 97.
 98. /**
 99.  * Deallocate
100.  *
101.  */
102. - (void)dealloc {
103.     // release all properties
104.     self.name = nil;
105.     self.favorite = nil;
106.     self.grades = nil;
107.     self.ohs = nil;
108.     self.students = nil;
109.
110.     [super dealloc];
111. }
112.
113. @end
114.
```

```objc
/**
 *  Student.h
 *  Section0App
 *  Tommy MacWilliam, 2011
 *
 */

#import <Foundation/Foundation.h>


@interface Student : NSObject {
    NSString *_name;
}

@property (copy, nonatomic, readwrite) NSString *name;

- (id)initWithName:(NSString *)newName;

@end

```

```objc
/**
 *  TF.h
 *  Section0App
 *  Tommy MacWilliam, 2011
 *
 */

#import <Foundation/Foundation.h>

@class Student;

@interface TF : NSObject {
    NSString *_name;
    NSMutableArray *_students;
    NSMutableArray *_ohs;
    Student *_favorite;
    NSDictionary *_grades;
}

@property (copy, nonatomic) NSString *name;
@property (retain, nonatomic) NSMutableArray *students;
@property (retain, nonatomic) Student *favorite;
@property (retain, nonatomic) NSDictionary *grades;
@property (retain, nonatomic) NSMutableArray *ohs;

- (id)initWithName:(NSString *)newName students:(NSMutableArray *)newStudents favorite:(Student *)newFavorite;
- (void)grade;
- (void)outputGrades;
- (void)addOfficeHour:(NSDate *)date;
- (void)outputOfficeHours;

@end

```