iOS Assignment 1

Identify as many problems as you can with the code below:

File T1.h

```
#import <Foundation/Foundation.h>

typedef void (^TestClassCallback)();

// Person is a subclass of NSManagedObject
@class Person;

@interface T1 : NSObject

- (void)doWorkWithPerson:(Person*)aPerson callback:(TestClassCallback)aCallback;

@end
```

File T1.m

```
#import "T1.h"
#import "Person.h"
#import "ProgressBar.h"

@implementation T1

static TestClassCallback savedCallback;

- (void)doWorkWithPerson:(Person*)aPerson callback:(TestClassCallback)aCallback
{
      savedCallback = aCallback;
      [self performSelectorInBackground:@selector(doVeryLongTask1:) withObject:aPerson];
}

- (void)doVeryLongTask1:(Person*)aPerson
{
      double p = 0.0;
      // Do some actions.
      // ...
      // ...
      [[ProgressBar instance] update:p];
      // Do more actions.
      // ...
      // ...
      [[ProgressBar instance] update:p];
      // Do final actions.
      (savedCallback)();
}

@end
```

iOS Assignment 2

For this assignment we have the Employee class. It has 3 properties:

- name
- birthYear
- salary

We also have EmployeeDirectory, which is a class that manages loading of employee records.

1) Create a selector in the Employee class that formats the salary of the employee according to the currency the salary is in and returns it as a string in readable format. You can use built-in libraries to show the proper currency symbols.
2) Create a View Controller that shows the list of employees (name, birthYear, and salary) loaded by the EmployeeDirectory. You should use the selector from the previous task to show the salary.
3) Make the View Controller have a button on its navigation toolbar that sorts the employee list by name without blocking the main thread. Use blocks.
4) Bonus points: don't use storyboards or XIBs.
5) Bonus points: use your own views with custom layout for each employee in the list.
6) Bonus points: where is the performance bottleneck in your code while scrolling your employee list (run on a device)? How could you solve it?

Employee.h

```
#import <Foundation/Foundation.h>

@interface Employee : NSObject

@property (readonly, copy) NSString* name;
@property (readonly) NSUInteger birthYear;
@property (readonly, copy) NSDecimalNumber* salary;

- (instancetype)initWithName:(NSString*)name birthYear:(NSUInteger)birthYear;

@end
```

Employee.m

```
#import "Employee.h"

static NSUInteger const kStartingSalary = 10000;
NSString* const kSalaryCurrency = @"EUR";

@implementation Employee

- (instancetype)initWithName:(NSString*)name birthYear:(NSUInteger)birthYear
{
    self = [super init];
    if(self)
    {
        _name = name;
        _birthYear = birthYear;
        _salary = [[NSDecimalNumber alloc] initWithUnsignedInteger:kStartingSalary];
    }
    return self;
}

@end
```

EmployeeDirectory.h

```
#import <Foundation/Foundation.h>

// notification posted when the directory finishes updating
extern NSString* const kEmployeeDirectoryDidUpdateNotification;
```

```
@interface EmployeeDirectory : NSObject

@property (readonly) NSArray* employees; // returns NSArray of Employee
@property (readonly) BOOL isUpdating;

- (void)update;

@end
```

EmployeeDirectory.m

```
#import "EmployeeDirectory.h"
#import "Employee.h"

NSString* const kEmployeeDirectoryDidUpdateNotification =
@"kEmployeeDirectoryDidUpdateNotification";

@implementation EmployeeDirectory

- (void)update
{
    if(_isUpdating == YES)
    {
        return;
    }
    _isUpdating = YES;

    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
        [self BA_doUpdateInBackground];
    });
}

#pragma mark - Privates

- (void)BA_doUpdateInBackground
{
    [NSThread sleepForTimeInterval:2];

    NSArray* name = @[@"Anne", @"Lucas", @"Marc", @"Zeus", @"Hermes", @"Bart", @"Paul", @"John",
@"Ringo", @"Dave", @"Taylor"];
    NSArray* surnames = @[@"Hawkins", @"Simpson", @"Lennon", @"Grohl", @"Hawkins", @"Jacobs",
@"Holmes", @"Mercury", @"Matthews"];

    NSUInteger amount = name.count*surnames.count;
    NSMutableArray* employees = [NSMutableArray arrayWithCapacity:amount];
    for(NSUInteger i=0; i<amount; i++)
    {
        NSString* fullName = [NSString stringWithFormat:@"%@ %@", name[random()%name.count],
surnames[random()%surnames.count]];
        [employees addObject:[[Employee alloc] initWithName:fullName birthYear:1997-random()%50]];
    }

    dispatch_async(dispatch_get_main_queue(), ^{
        [self BA_updateDidFinishWithResults:employees];
    });
}

- (void)BA_updateDidFinishWithResults:(NSArray*)results
{
    _employees = results;
    _isUpdating = NO;

    [[NSNotificationCenter defaultCenter]
postNotificationName:kEmployeeDirectoryDidUpdateNotification object:self];
}

@end
```