[Objective-C Exercise] answered by BoYoung

16. With your own words, explains what **ARC** is and what advantages it brings to Objective-C. Give details about it.

Before explaining ARC, I searched about reference counting. Every object has 'retainCount' and a developer can manage when each object is taken away in memory.
(alloc +1, retain +1, release -1, autorelease -1)
=> When the retain count is '0', the object is taken away in memory. (not right now)
=> In this case, more code is needed and can be inefficient sometimes.
=> Actually, 'Manual reference counting' is not used much after adopting ARC.

It happens because Objective-C doesn't have 'garbage collector'. ARC has a roll same as 'garbage collector' in Java.

ARC (Automatic Reference Counting) manages 'reference counting' automatically and it can manage memory efficiently.
=> ARC is only in objects of Objective-C and an attribute of Compiler.
=> ARC plays roles in 'retain','release','autorelease' and 'dealloc'. It means developers don't need to care about 'retain/release code'.

So simpler and less bug codes can be written and developer can manage codes more efficiently as well.

17. With your own words, explain in details the main differences between **NSArray** and **NSDictionary**. How about **NSMutableArray** and **NSMutableDictionary**?

It was better for me to understand those when comparing to Java.

Objective-C	JAVA
NSArray 1) We can store each value like array of Java. 2) Any type of objects can stores in NSArray. : NSString / NSNumber / any object	**Array** 1) It can have same type data. EX.) int[] test = new int[7] => all data should be integer.

NSMutableArray
1) The number of objects is changeable.
: An undefined number of data can be added.
EX.)
NSMutableArray* mutuableArray = [[NSMutabalArray alloc] initWithObjects:@"A"];
[mutuableArray addObject:@"B"];
[mutuableArray addObject:1];
…
	arrayList
	1) The number of objects is changeable.
	: An undefined number of data can be added.
	EX.)
	ArrayList<Integer> arrayList = new ArrayList<>();
	arrayList.add(1);
	arrayList.add(2);
	…

NSDictionary 1) It is similar to map in Java. : It has “key” value and “object” value. : In array, we get the value by using the index. In Dictionary, we get the object by “Key”. 2) It cannot be changed. : We can not add and remove objects.	**Map** 1) When the data is stored, “key” and “value” both are stored. : We can get the value by key.
NSMutableDictionary 1) It plays a same roll in NSDictionary. However, objects can be added and removed.	

18. With your own words, explain in details the main differences between **NSString** and **char***.

- The main difference is what they stores.
: (NSString) stores and handles the pointer of string objects. / immutable.
: (char*) stores and handles the pointer of each char (array). / mutable.

- <NSString>
: is only used in Objective-C.
: is one of the class, a class object is created by reference type.
: has related methods like String class of Java.
: We should put “@“(Objective-C String) before the object.

19. With your own words, explain in details what an event-driven app is. Given relevant examples about it.

1) Event-driven app is an application that works by some events. (The flow of program depends upon the events.)
2) There are many kinds of events.
: User’s actions (typing keyboard / clicking mouse / touching screen)
: Receiving data from network
: Life cycle
: and so on.
3) To get and receive results of events, event listener is needed.

In Objective-C,
: We can handle events in many ways.
  - [UIApplication / UIApplicationMain] handle the event by life-cycle.
  - [NSView / NSWindow / NSApplication / NSEvent] handle mouse events, key events, and so.

: For example, We can handle arrow keys in keyboard when we make a game.

```
- (void) keyDown : (NSEvent *) theEvent
 {
        NSString* keys = [theEvent characters];

        for (int i = 0 ; i < keys.length ; ++i)
        {
                switch (keys characterAtIndex: 1)
                {
                        case NSLeftArrowFunctionKey:
                        {
                                doSomething();
                                break;
                        }
                        case NSRightArrowFunctionKey:
                        {
                                doSomething();
                                break;
                        }
                        case NSUpArrowFunctionKey:
                        {
                                doSomething();
                                break;
                        }
                        case NSDownArrowFunctionKey:
                        {
                                doSomething();
                                break;
                        }
                        default:
                        {
                                [super keyDown:theEvent];
                                break;
                        }
                }
        }
```

20. With your own words, explain in details how to avoid a strong reference cycle. Give relevant examples about it.

Strong Reference Cycle: When 2 or more classes refers each other or one another, it means they are holding forever. So the number of references can never be ‘0’. It makes memory waste.

To avoid this, we can use ‘weak’ or ‘unknown’ types for references.
In Exercise 13-14, ’weak’ property modifier is used for Computer class, Keyboard class and Mouse class.

Weak and Unknown both don’t increase the retain count of the object.
In swift, Unknown is not optional and always has the value.
Therefore,
- When it doesn’t matter if the value is nil, it is better to use Weak.
- When the value is needed in whole life-cycle, it is better to use Unknown.