Swift Advance: Assignment

Q1. What is extension ?

Ans:- Extensions add new features to an existing class, structure,enumeration, or protocol type.This includes the ability to extend types for which you do not have access to the original source code.

```
Syntax:-  extension someType
          {
                  //adding some new functionality over here
          }
```

Q5. What is Generics?

Ans:- Generics in swift enables us to write flexible,reusable functions and types that can work with any type,subject to requirements that we define.We can write code that avoids duplication and expresses its intent in a clear, abstracted manner.

Q6. Explain Generics with an Example?

Ans:- Let suppose we have a function to swap two numbers as-

```
 Func swapTwoInts(_ a : inout Int, _ b : inout Int)
{
      let temporaryA = a
      a = b
      b = temporaryA
}
```

This Function is very useful but can only swap two Integer numbers.But if we want to swap two String or Double values we have to write two seperate functions.

- On the other hand generic function can work with any type as shown below.

```
Func swapTwovalues<T>(_ a : inout T, _ b : inout T)
{
         let temporaryA = a
        a = b
        b = temporaryA
}
```

In this generic version of the function we use placeholder <T> for any time instead of any actual type.In this way we have generalized our given function, but we should note that the given function have both arguments of type <T> so both of them should be of same type.

Q7. Difference between map() and compactMap() with example?

Ans:- map() - It takes each value from the collection such as array and run it through closure or any given function to transform it.

```
eg :-  let numbers = [1,2,3,4,5]
         Let doubled = numbers.map{$0 * 2}
         print(doubled)
         Output:- [1,4,6,8,10]
```

compactMap() - On the other hand compactMap() gives us advantage over simple map() when we work with optionals or nil values that might be present in our collections such as array. compactMap() ignores the nil value and final output does not contain any nil value which is not possible in simple map().

```
eg:- let let numbers = [“42”,”19”,”notAnumber”]
       Let ints = numbers.compactMap{ Int( $0 ) }
       print(ints)
       Output:- [42,19]
```

So in this output the last value of array evaluates to nil which is ignored.