BITS F301 – Principles of Programming Languages

Assignment 3

Function Codes in Scala

1. Sagar Gupta - 2014A7PS030H
2. Ravi Teja – 2014A7PS196H
3. Anirud Samala - 2014A7PS027H
4. Dhawal - 2014A7PS112H

Functions Shown

Function to find kth largest element in a list -

```
def findKth[myList](k:Int, myList: Array[Int]):Int =  {
        var arrLen = myList.length;
        for(i<-0 to (arrLen - 1)) {
                for(j<-0 to (arrLen - 2)) {
                        if(myList(j) > myList(j+1)) {
                                var temp = myList(j);
                                myList(j) = myList(j+1);
                                myList(j+1) = temp;
                        }
                }
        }
        //sorting array ascending order
        println(k+"th largest element is: "+myList((myList.length - k)));
        return myList((myList.length - k));
}
```

Function to find kth largest element in a list -

```
def bubSort[myList](myList: Array[Int]):Boolean =  {
        var arrLen = myList.length;
        for(i<-0 to (arrLen - 1)) {
                for(j<-0 to (arrLen - 2)) {
                        if(myList(j) > myList(j+1)) {
                                var temp = myList(j);
                                myList(j) = myList(j+1);
                                myList(j+1) = temp;
                        }
                }
        }
        return true;
}
```

Function to find average of nos in a list -

```
def rotate[A](n: Int, ls: List[A]): List[A] = {
        val nBounded = if (ls.isEmpty) 0 else n % ls.length
        if (nBounded < 0) rotate(nBounded + ls.length, ls)
        else (ls drop nBounded) ::: (ls take nBounded)
        }
```

Function to perform binary search over a list -

```
def binary_search(target:Int, l:List[Int]) = {
        def recursion(low:Int, high:Int):Option[Int] = (low+high)/2 match{
          case _ if high < low => None
          case mid if l(mid) > target => recursion(low, mid-1)
          case mid if l(mid) < target => recursion(mid+1, high)
          case mid => Some(mid)
        }
        recursion(0,l.size - 1)
      }
```

Function to rotate over a list -

```
def rotate[A](n: Int, ls: List[A]): List[A] = {
      val nBounded = if (ls.isEmpty) 0 else n % ls.length
      if (nBounded < 0) rotate(nBounded + ls.length, ls)
      else (ls drop nBounded) ::: (ls take nBounded)
  }
```

Function to find if given string is palindrome -

```
def isPalindrome[A](l: List[A]):Boolean = {
        l == l.reverse
}
```

Function to find reverse of a list -

```
def reverse[A](ls: List[A]) = {
      ls.reverse
}
```