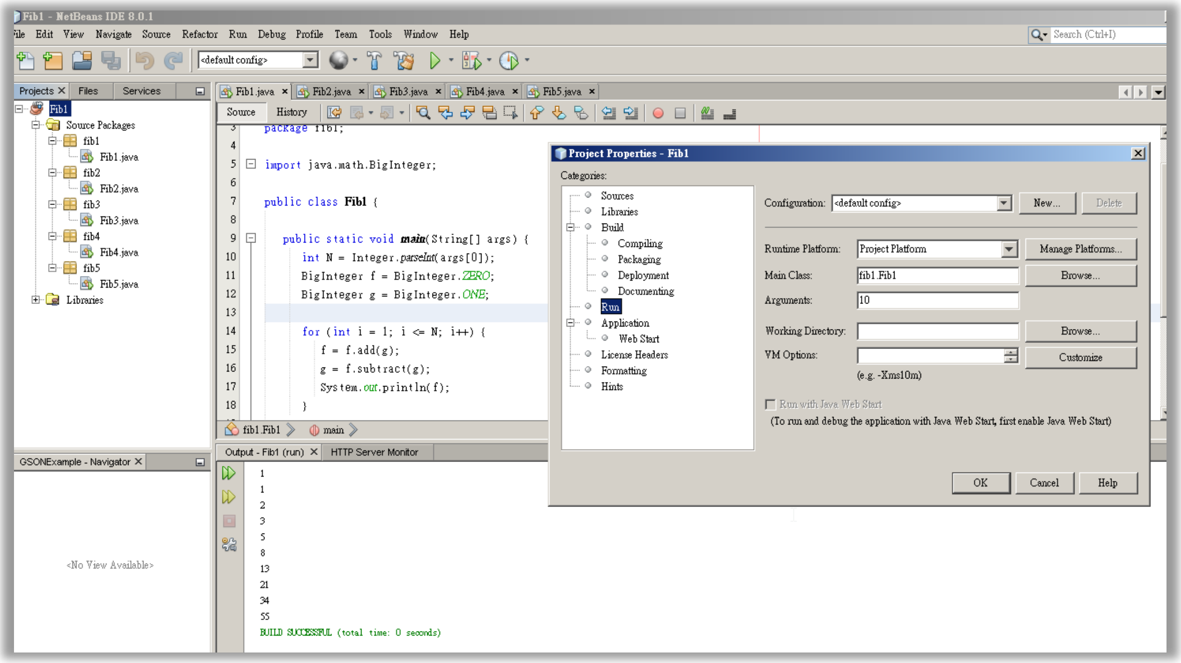
Name: Ho Ping Chong
Student no: 1155057016

GNBF5010-Assigment 3
Question:
Write a program to print Fibonacci sequence. The length of output sequence is specified by the first command line parameter

Answer:
The Fibonacci Sequence is defined as the sequence of numbers with f0 = 0, f1 = 1, and fn = fn–1 + fn −2 for n ≥ 2. There are numerous algorithms on Fibonacci sequence (http://www.ics.uci.edu/~eppstein/161/960109.html). From simply For loop iteration, recursive function, memorized recursion, and complicate algorithms such as matrix exponentiation and fast doubling (http://nayuki.eigenstate.org/page/fast-fibonacci-algorithms).
.
For loop iteration:

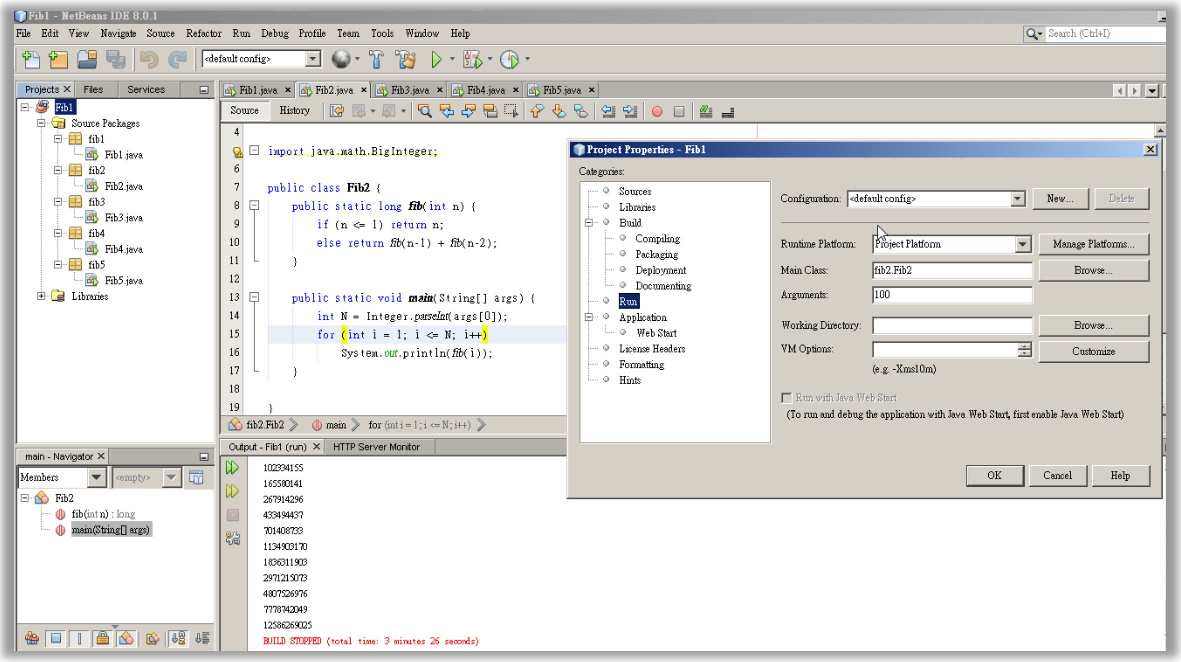
```
/*  For loop iteration */

package fib1;

import java.math.BigInteger;

public class Fib1 {

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      BigInteger f = BigInteger.ZERO;
      BigInteger g = BigInteger.ONE;

      for (int i = 1; i <= N; i++) {
         f = f.add(g);
         g = f.subtract(g);
         System.out.println(f);
      }
   }
}
```



Recursive function:

The amount of time to compute F(n) is proportional to the resulting value itself, which grows exponentially.

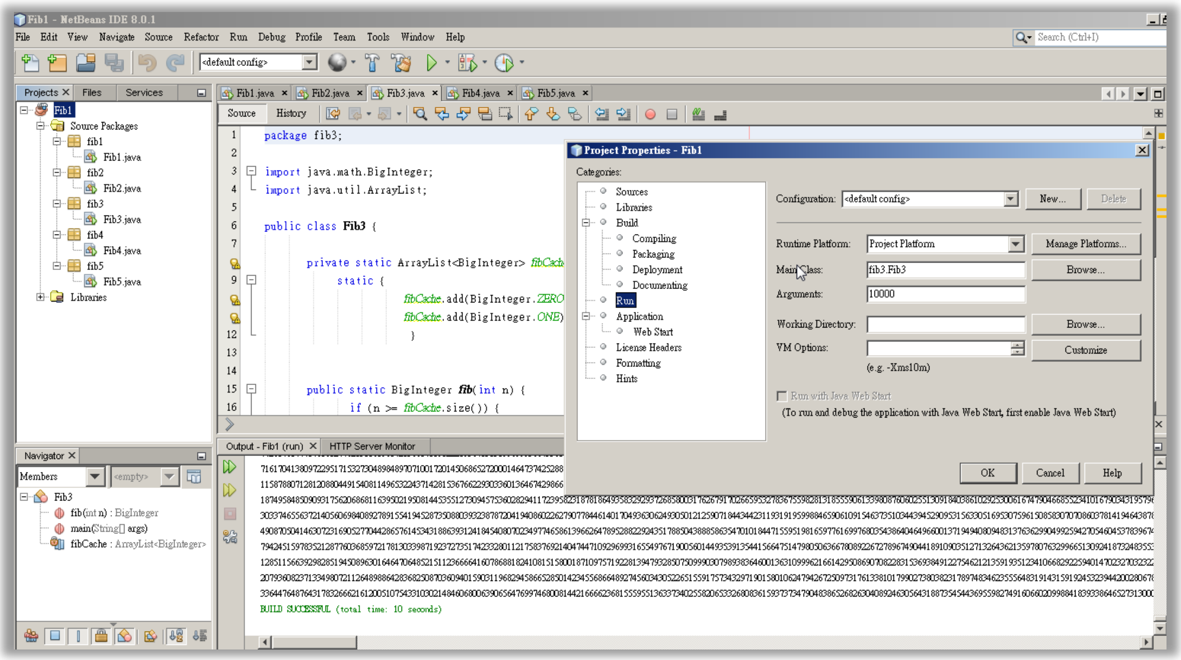
```
/* Recursive function with inefficient and illustrate the performance bug */

package fib2;

public class Fib2 {
    public static long fib(int n) {
        if (n <= 1) return n;
        else return fib(n-1) + fib(n-2);
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            System.out.println(fib(i));
    }

}
```



Memorized recursion:

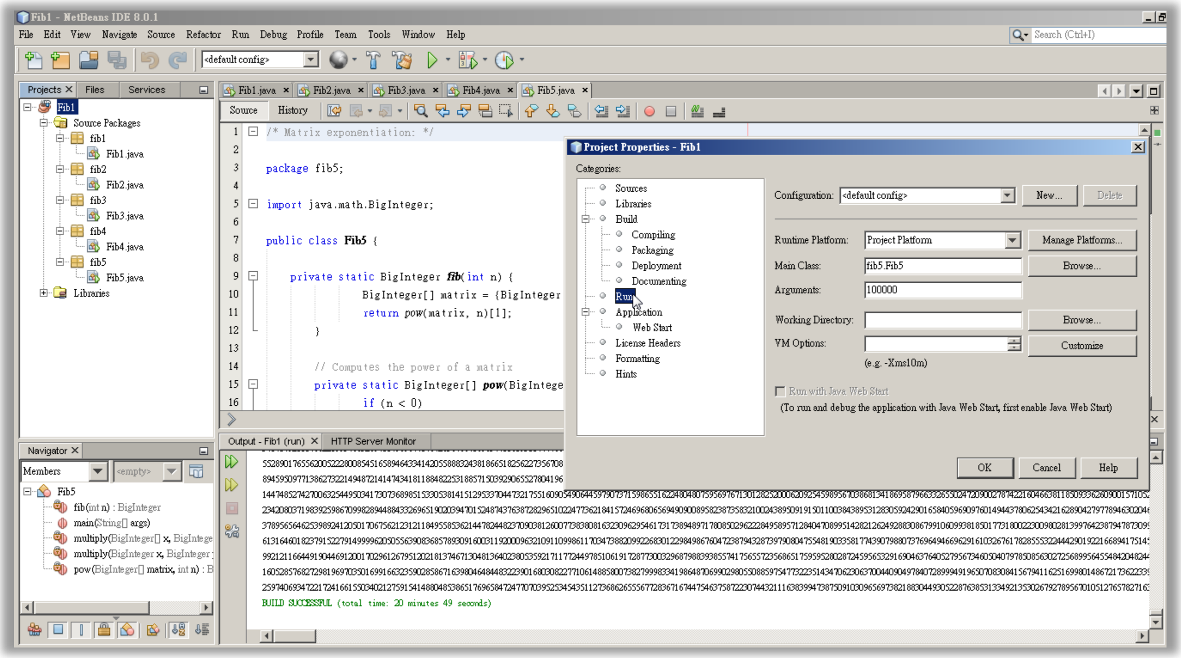
It is similar to the basic recursive function but a new fibonacci number is only computed if the cache does not already contain the necessary value. Newly computed fibonacci numbers are added to the cache, so that they become available during later calls to fib().

```
package fib3;

import java.math.BigInteger;
import java.util.ArrayList;

public class Fib3 {

    private static ArrayList<BigInteger> fibCache = new ArrayList<BigInteger>();
      static {
            fibCache.add(BigInteger.ZERO);
            fibCache.add(BigInteger.ONE);
             }


    public static BigInteger fib(int n) {
        if (n >= fibCache.size()) {
           fibCache.add(n, fib(n-1).add(fib(n-2)));
        }
        return fibCache.get(n);
    }

      public static void main(String[] args) {
          int N = Integer.parseInt(args[0]);
             for (int i = 1; i <= N; i++)
                System.out.println(fib(i));
      }

}
```



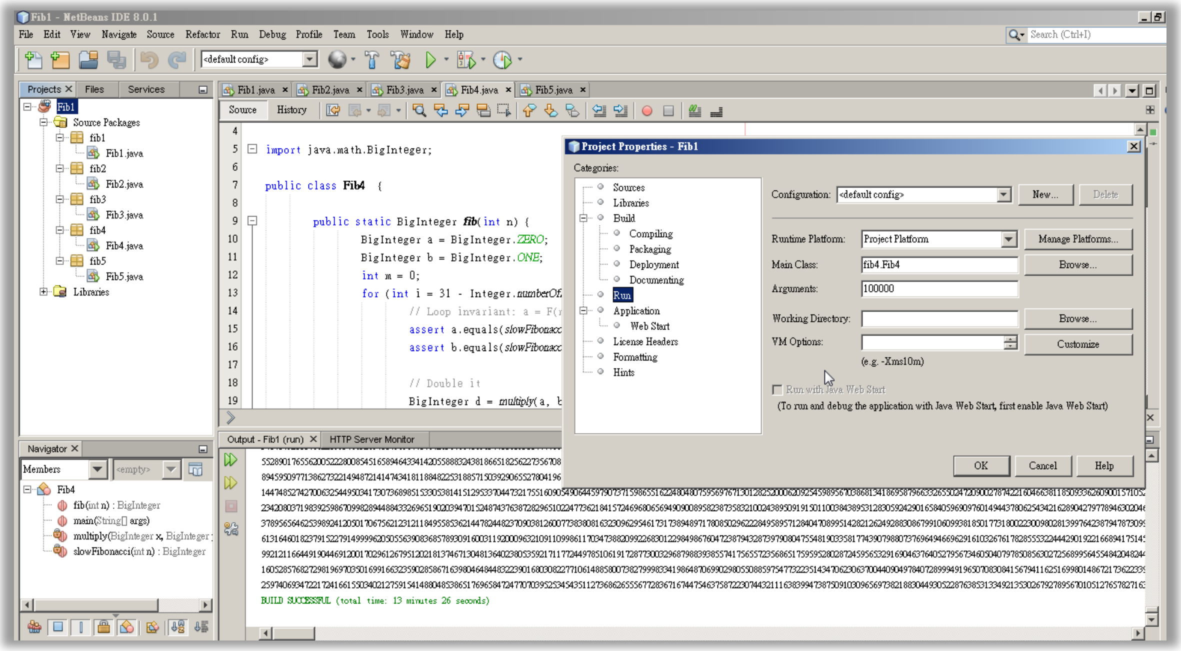
Matrix exponentiation:

```
/* Matrix exponentiation: */

package fib5;

import java.math.BigInteger;

public class Fib5 {

    private static BigInteger fib(int n) {
                BigInteger[] matrix = {BigInteger.ONE, BigInteger.ONE, BigInteger.ONE, BigInteger.ZERO};
                return pow(matrix, n)[1];
        }

        // Computes the power of a matrix
        private static BigInteger[] pow(BigInteger[] matrix, int n) {
                if (n < 0)
                        throw new IllegalArgumentException();
                BigInteger[] result = {BigInteger.ONE, BigInteger.ZERO, BigInteger.ZERO, BigInteger.ONE};
                while (n != 0) {  // Exponentiation by squaring
                        if (n % 2 != 0)
                                result = multiply(result, matrix);
                        n /= 2;
                        matrix = multiply(matrix, matrix);
                }
                return result;
        }

        // Multiplies two matrices
        private static BigInteger[] multiply(BigInteger[] x, BigInteger[] y) {
                return new BigInteger[] {
                        multiply(x[0], y[0]).add(multiply(x[1], y[2])),
                        multiply(x[0], y[1]).add(multiply(x[1], y[3])),
                        multiply(x[2], y[0]).add(multiply(x[3], y[2])),
                        multiply(x[2], y[1]).add(multiply(x[3], y[3]))
                };
        }

           private static BigInteger multiply(BigInteger x, BigInteger y) {
                return x.multiply(y);
        }

            public static void main(String[] args) {
                int N = Integer.parseInt(args[0]);
                  for (int i = 1; i <= N; i++)
                  System.out.println(fib(i));
             }
}
```


Fib1 - NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects Files Services
Fib1
Source Packages
fib1
Fib1.java
fib2
Fib2.java
fib3
Fib3.java
fib4
Fib4.java
fib5
Fib5.java
Libraries
Fib1.java Fib2.java Fib3.java Fib4.java Fib5.java
Source History
/* Matrix exponentiation: */
package fib5;
import java.math.BigInteger;
public class Fib5 {
private static BigInteger fib(int n) {
BigInteger[] matrix = {BigInteger
return pow(matrix, n)[1];
}
// Computes the power of a matrix
private static BigInteger[] pow(BigIntege
if (n < 0)
Project Properties - Fib1
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Configuration: <default config>
New...
Delete
Runtime Platform: Project Platform
Manage Platforms...
Main Class: fib5.Fib5
Browse...
Arguments: 100000
Working Directory:
Browse...
VM Options:
Customize
(e.g. -Xms10m)
Run with Java Web Start
(To run and debug the application with Java Web Start, first enable Java Web Start)
OK
Cancel
Help
Navigator
Members
<empty>
Fib5
fib(int n) : BigInteger
main(String[] args)
multiply(BigInteger[] x, BigInteger
multiply(BigInteger x, BigInteger
pow(BigInteger[] matrix, int n) : B
Output - Fib1 (run)
HTTP Server Monitor
BUILD SUCCESSFUL (total time: 20 minutes 49 seconds)

Fast doubling:

```
/* Fast doubling */

package fib4;

import java.math.BigInteger;

public class Fib4  {

	public static BigInteger fib(int n) {
		BigInteger a = BigInteger.ZERO;
		BigInteger b = BigInteger.ONE;
		int m = 0;
		for (int i = 31 - Integer.numberOfLeadingZeros(n); i >= 0; i--) {
			// Loop invariant: a = F(m), b = F(m+1)
			assert a.equals(slowFibonacci(m));
			assert b.equals(slowFibonacci(m+1));

			// Double it
			BigInteger d = multiply(a, b.shiftLeft(1).subtract(a));
			BigInteger e = multiply(a, a).add(multiply(b, b));
			a = d;
			b = e;
			m *= 2;
			assert a.equals(slowFibonacci(m));
			assert b.equals(slowFibonacci(m+1));

			// Advance by one conditionally
			if (((n >>> i) & 1) != 0) {
				BigInteger c = a.add(b);
				a = b;
				b = c;
				m++;
				assert a.equals(slowFibonacci(m));
				assert b.equals(slowFibonacci(m+1));
			}
		}
		return a;
	}

	private static BigInteger slowFibonacci(int n) {
		BigInteger a = BigInteger.ZERO;
		BigInteger b = BigInteger.ONE;
		for (int i = 0; i < n; i++) {
			BigInteger c = a.add(b);
			a = b;
			b = c;
		}
		return a;
	}

		private static BigInteger multiply(BigInteger x, BigInteger y) {
		return x.multiply(y);  // Replace this line with Karatsuba multiplication, etc. if available
	}


	    public static void main(String[] args) {
	        int N = Integer.parseInt(args[0]);
	          for (int i = 1; i <= N; i++)
	          System.out.println(fib(i));
	}

 }
```


Fib1 - NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Project Properties - Fib1
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Configuration: <default config>
Runtime Platform: Project Platform
Main Class: fib4.Fib4
Arguments: 100000
Working Directory:
VM Options:
(e.g. -Xms10m)
Run with Java Web Start
(To run and debug the application with Java Web Start, first enable Java Web Start)
OK
Cancel
Help
BUILD SUCCESSFUL (total time: 13 minutes 26 seconds)

Performance:

Algorithm	Time (N=10)	Time (N=100)	Time (N=1000)	Time (N=10000)	Time(N=100000)
Iteration	0sec	0sec	0sec	11sec	12mins 52sec
Recursion	0sec	> 60sec	NA	NA	NA
Memorized recursion	0sec	0sec	0sec	10sec	12mins 10sec
Double fasting	0sec	0sec	0sec	10sec	13mins 26sec
Matrix exponentiation	0sec	0sec	0sec	10sec	20mins 49sec

Summary:

Simple iterative algorithm consume $O(n^2)$ as the total time that takes. Since we only need to keep two numbers in memory at once, O(n) is the space needed. (http://pages.cs.wisc.edu/~mhock/SSL/fibcalc.pdf)

Recursive algorithm is extremely slow. It uses O(n) stack space and O(ϕn) time, where ϕ=5√+12 (the golden ratio). The amount of time to compute F(n) is proportional to the resulting value itself, which grows exponentially. It is always beaten by simple iterative algorithm.

Memorized recursion or dynamic programming has reasonable performance and easily to be understand. It is similar to the basic recursive function but a new fibonacci number is only computed if the cache does not already contain the necessary value. If we've already computed F(k−2) and F(k−1), then we can add them to get F(k). Next, we add F(k−1) and F(k) to get F(k+1). The algorithm calculates each fibonacci number in sequence. This algorithm uses *O*(*n*) time.

Matrix exponentiation is using matrix multiplication. This takes time O(n) which faster than simple recursion algorithm and probably slower than memorized recursion algorithm. The time spent in a call to pow (O(n)) plus the time in each recursive call.

Fast doubling algorithm. It Just like in the Matrix exponentiation method, the fast doubling algorithm runs in O(M(n) Log(n)), where M(n) is the time for the multiplication of two numbers with n digits. The difference between this method and the closed-form matrix formula is the doubling algorithm requires fewer redundant steps if one avoids to recompute an already computed Fibonacci number (memorized recusion), making this algorithm one of the fastest methods to compute the Fibonacci sequence. (http://web.ist.utl.pt/~catarina.p.moreira/programming/FibDoubling.html)

Running time isn't the only thing we should concern since we will often analyze the amount of memory used by a program. If a program takes a lot of time, it just waits longer for the result. However if a program takes a lot of memory, it may fail to run, so this is an important parameter to understand.

The matrix method of generating Fibonacci numbers is more efficient than the simple iterative algorithm. It works with numbers consisting of hundreds of bits or more. For smaller numbers, the simplicity of the iterative algorithm is preferable. (http://pages.cs.wisc.edu/~mhock/SSL/fibcalc.pdf) The fast doubling is theoretically faster than matrix exponentiation through memorization. (http://web.ist.utl.pt/~catarina.p.moreira/programming/FibDoubling.html)

In our findings, the memorized recursion is the fastest, the next is simple iteration, fast doubling and matrix exponentiation in order. Simple recursion is not functional when larger Fibonacci number.

I am submitting the assignment for:

x□ an individual project or

□ a group project on behalf of all members of the group. It is hereby confirmed that the submission is authorized by all members of the group, and all members of the group are required to sign this declaration.

I/We declare that the assignment here submitted is original except for source material explicitly acknowledged, the piece of work, or a part of the piece of work has not been submitted for more than one purpose (i.e. to satisfy the requirements in two different courses) without declaration, and that the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted. I/We also acknowledge that I am/we are aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website http://www.cuhk.edu.hk/policy/academichonesty/. In the case of a group project, we are aware that each student is responsible and liable to disciplinary actions should there be any plagiarized contents in the group project, irrespective of whether he/she has signed the declaration and whether he/she has contributed directly or indirectly to the plagiarized contents.

It is also understood that assignments without a properly signed declaration by the student concerned and in the case of a group project, by all members of the group concerned, will not be graded by the teacher(s).

Ho Ping Chong	7 Nov 2014
Signature(s)	Date
Ho Ping Chong	1155057016
Name(s)	Student ID(s)
GNBF5010	Introduction to programming
Course code	Course title