جامعة فيلادلفيا Philadelphia University	اسم النوذج: وصف المادة	QFO-AP-FI-MO02
	الجهة المصدرة: كلية تكنولوجيا المعلومات	رقم الإصدار: 1 (Revision)
	الجهة المدققة: عمادة التطوير والجودة	التاريخ: 2017/11/05
		عدد صفحات النموذج:

Course Title: **Programming Fundamentals(1)**	**Course code:** 750113
Course Level: 1	**Course prerequisite (s) and/or corequisite(s):**
Lecture Time:	**Credit hours: 3**

Academic Staff Specifics				
Name	**Rank**	**Office No. and Location**	**Office Hours**	**E-mail Address**

Course/Module Description:
This module focuses on problem solving strategies and the use of algorithmic language to describe such problem solving. It introduces the principles of procedural programming, data types, control structures, data structures and functions, data representation on the machine level. Various problems are considered to be solved using C-like procedural programming language.

Course/Module Objectives:
This module aims to introduce computer programming and emphasis in problem solving on the fundamentals of structured design using the principles of Top Down problem solving strategy (divide and conquer). This includes development, testing, implementation, documentation.
The module also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers.

Course/ module components

- **Textbook:**
  - P. Deitel & H. Deitel, **C++ How to program,** Pearson Education Limited, 2013.

- **Supporting material(s):** Lectures handouts
  Introduction to Computer Science and Programming (Spring 2011) (MIT)
  Introduction to C++ (MIT)

Teaching methods:
Duration: 16 weeks, 80 hours in total

Lectures: 32 hours (2 hours per week),
Tutorials: 16 hours (1 per week),
Laboratories: 32 hours, 2 per week

Learning outcomes

A- Knowledge and understanding

A2. Know & understand a wide range of principles and tools available to the software developer, such as design methodologies, choice of algorithm, language, software libraries and user interface technique:

A4. Know & understand a wide range of software and hardware used in development of computer systems

A5. Know & understand the professional and ethical responsibilities of the practising computer professional including understanding the need for quality, security, and computer ethics.

B- *Intellectual skills (thinking and analysis).*

B1. Analyze a wide range of problems and provide solutions through suitable algorithms, structures, diagrams, and other appropriate methods

B4. Practice self learning by using the e-courses

C- *Practical skills*

C3. Work effectively with and for others.

C4. Strike the balance between self-reliance and seeking help when necessary in new situations

C5. Display personal responsibility by working to multiple deadlines in complex activities

D- *Transferable Skills*

D2. Prepare and deliver coherent and structured verbal and written technical reports.

D4. Use the scientific literature effectively and make discriminating use of Web resources

D5. Design, write, and debug computer programs in appropriate languages

Learning outcomes achievement

- Development: A2, A4, and A5 are developed through the lectures and laboratory sessions.
  B1, D5, C3, and C4 are developed trough Tutorials and Lab sessions,
  B4, D2, D4, D5, and C5 are developed through Homework

- Assessment : A2, A4, A5, B1, D5, and C4 and are assessed through Quizzes, written exams, and Practical Works Exams.
  B4, D2, D4, D5, and C5 are assessed through Homework Exam.

Assessment instruments

Allocation of Marks	
Assessment Instruments	**Mark**
First examination	**15%**
Second examination	**15%**
Final examination	**40%**
Lab works, Quizzes, and tutorial contributions	**30%**
Total	**100%**

Course/Module Academic Calendar

Week	Basic and support material to be covered	Homework/reports and their due dates
(1)	***Problem Solving (an Object approach)***: Process, Analyze (requirement, Design algorithm, Tracing algorithm). Example	
(2)	**Tutorial 1** Elementary algorithm object design and tracing (natural languages)	**Lab work #0** Introduction to lab regulation and process
(3)	***Data Definition Structures***: Simple data types, constants, variables, and its internal representations. Compiler directives , namespace, const	**Lab work #1**
(4)	***Data Definition Structures*** :Expressions: Arithmetic, Logical; Precedence rules; I/O manipulators**,** I/O formatting flags **Tutorial 2**	**Lab work #2**
(5)	***Problem Analysis***: ***An Object approach)***. Algorithm discovery, Algorithm design strategies**,** Stepwise refinement**,** Control requirements Examples.	**Lab work #3** ()
(6)	**Tutorial 3** Min in a set, Max in a set, Sum of a set, Implementing algorithm, Conclusion,	**Lab work #6** ()
(7)	***Control Structures***: Sequencing; Input and output statements; Assignment statement;	**Lab work #7** ()
(8) First examination	***Control Structures***: Examples **Tutorial 4**	**Lab work #8** ()
(9)	***Control Structures***: Selection: one-way (if .. then), two-way (if .. then .. else), multi-way (nested if) (switch);	**Lab work #9** ()
(10)	**Tutorial 5** Counting even numbers, counting 'XE' sequences, counting word starting by 'LE' in a string,	**Lab work #10** ()
(11)	***Control Structures***: Repetition (while structure); **Tutorial 6.** Searching a value, process on a set delimited by a sentinel value	**Lab work #11** ()
(12) Second examination	***Control Structures***: Repetition (do while, for); **Tutorial 7**. Redo the same problems and others with do while and for.	**Lab work #12** ()
(13)	***Control Structures***: use different control structures. **Tutorial 8**. Several exercises combining all the control structures.	**Lab work #13** ()
(14)	Enumeration data types**, user defined types, scope of names, type casting**	**Lab work #14** ()
(15)	Practical Exam	**Lab work #15** ()
(16) Final Examination	Final Exam	**Lab work #16** (Revision)

Expected workload:
On average students need to spend 3 hours of study and preparation for each 50-minute lecture/tutorial.

Attendance policy:
Absence from lectures and/or tutorials shall not exceed 15%. Students who exceed the 15% limit without a medical or emergency excuse acceptable to and approved by the Dean of the relevant college/faculty shall not be allowed to take the final examination and shall receive a mark of zero for the course. If the excuse is approved by the Dean, the student shall be considered to have withdrawn from the course.

Module references
Students will be expected to give the same attention to these references as given to the Module textbook(s)

1. D.S. Malik , Thomson, C++ Programming: From Problem Analysis to Program Design, Sixth Edition, Course Technology, 2012.
2. Friedman Frank and Koffman Elliot B., "*Problem Solving, Abstraction and Design using C++*", Pearson Education , 2011.
3. A. Lambert Kenneth and Nance Douglas W., "*Understanding Programming and Problem Solving With C++*", PWS Publishing Company, Fourth Edition. 1996
4. Forouzan, B. A. & R. F. Gilberg. *"Computer Science: A Structured Programming Approach using C"*, Second Edition, Pacific Grove, CA: Brooks/Cole, 2001
5. Bruce Eckel, *"Thinking in C++"*, Second Edition, Prentice Hall, 2000.
6. Herbert Schildt, *"Teach Yourself C++"*, Third Edition, McGraw-Hill. 1998.

Website(s):

- www.cee.hw.zc.uk/~pjbk/pathways/cpp1/cpp1.html
- www.edm2.com/0507/introcpp1.html
- www.doc.ic.ac.uk/~wjk/C++intro
- www.cprogramming.com/tutorial.html
- www.cs.umd.edu/users/cml/cstyle/ellemtel-rules.html
- www.deakin.edu.au/~agoodman/Ctutorial.html
- www.tldp.org/howto/c++programming.howto.html
- www.vb-bookmark.com/cpptutorial.html

DOCUMENTATION FOR PROGRAMS:
(All programming assignments must include at least the following comment lines)

/***TASK:**	Identify what the program will accomplish	*/
/*	*/	
/***WRITTEN BY**:		*/
/*	*/	
/***DATE**:	List creation & modification dates	*/
/*	*/	
/***VARIABLES:**	List and give what each represents	*/
/*	*/	
/***INPUT:**	Identify the input parameters: Give examples	*/
/*	*/	
/***OUTPUT:**	Identify the expected output: Give examples	*/
/*	*/	

```
/*ALGORITHM:      Briefly describe the algorithm used*/
#include <stdio.h>
main ( )
{    …  }
```

(If your program includes any function modules, each function needs to be documented)

```
/*TASK:           Identify what the function accomplishes                    */
/*                                                                           */
/*DATE:           List creation and modification dates                       */
/*                                                                           */
/*WRITTEN BY:                                                                */
/*                                                                           */
/*VARIABLES:      List names and what each represents                        */
/*                                                                           */
/*INPUT:          Identify the input parameters, if any. Give examples       */
/*                                                                           */
/*OUTPUT:         Identify the output. Give examples                         */
/*                                                                           */
/*ALGORITHM:      Briefly describe the algorithm used                        */
 int function1( )
   {   …  }
```