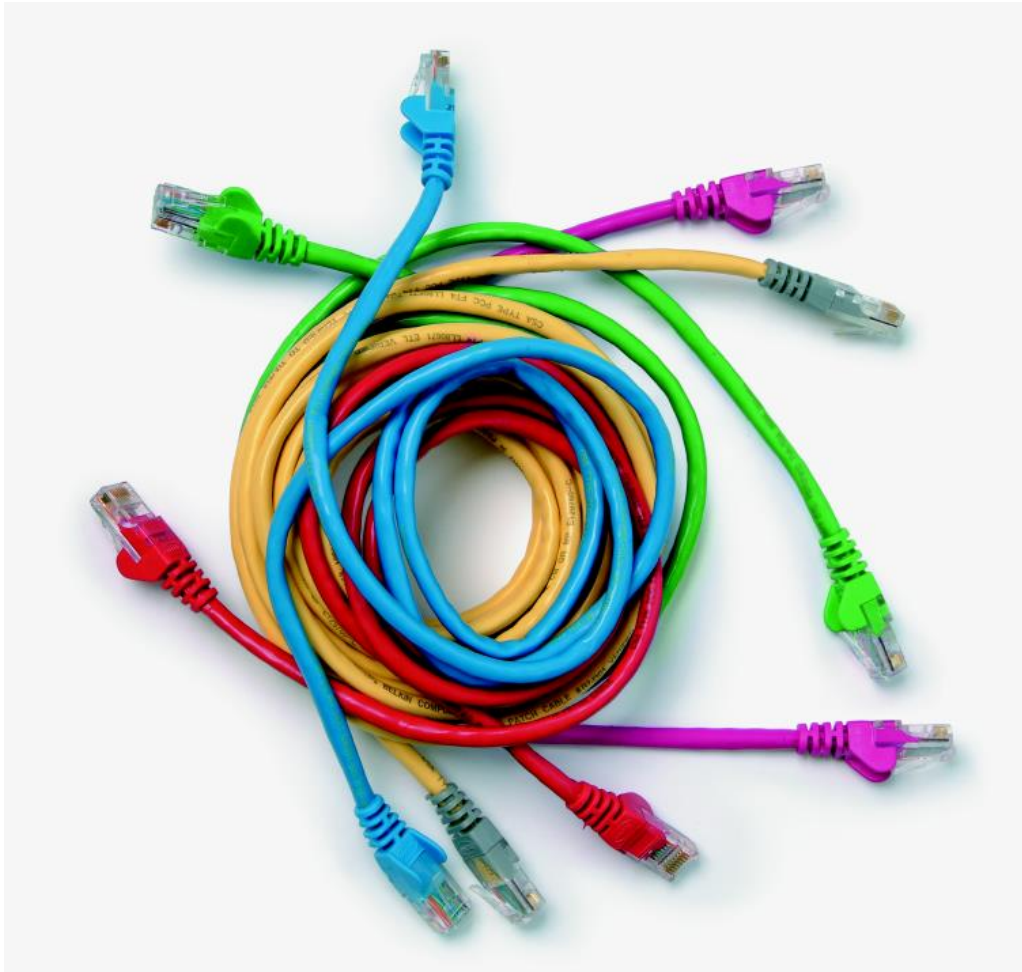


# Level 3 Software design fundamentals

**(7540-033/7630-348)**

**Systems and Principles**  
**Assignment guide for Candidates**  
Assignment C



## **About City & Guilds**

City & Guilds is the UK's leading provider of vocational qualifications, offering over 500 awards across a wide range of industries, and progressing from entry level to the highest levels of professional achievement. With over 8500 centres in 100 countries, City & Guilds is recognised by employers worldwide for providing qualifications that offer proof of the skills they need to get the job done.

## **City & Guilds Group**

The City & Guilds Group includes City & Guilds, ILM (the Institute of Leadership & Management) which provides management qualifications, learning materials and membership services, NPTC which offers land-based qualifications and membership services, and HAB (the Hospitality Awarding Body). City & Guilds also manages the Engineering Council Examinations on behalf of the Engineering Council.

## **Equal opportunities**

City & Guilds fully supports the principle of equal opportunities and we are committed to satisfying this principle in all our activities and published material. A copy of our equal opportunities policy statement is available on the City & Guilds website.

## **Copyright**

The content of this document is, unless otherwise indicated, © The City and Guilds of London Institute 2008 and may not be copied, reproduced or distributed without prior written consent.

However, approved City & Guilds centres and learners studying for City & Guilds qualifications may photocopy this document free of charge and/or include a locked PDF version of it on centre intranets on the following conditions:

- centre staff may copy the material only for the purpose of teaching learners working towards a City & Guilds qualification, or for internal administration purposes
- learners may copy the material only for their own use when working towards a City & Guilds qualification

The *Standard Copying Conditions* on the City & Guilds website also apply.

Please note: National Occupational Standards are not © The City and Guilds of London Institute. Please check the conditions upon which they may be copied with the relevant Sector Skills Council.

## **Publications**

City & Guilds publications are available on the City & Guilds website or from our Publications Sales department at the address below or by telephoning +44 (0)20 7294 2850 or faxing +44 (0)20 7294 3387.

Every effort has been made to ensure that the information contained in this publication is true and correct at the time of going to press. However, City & Guilds' products and services are subject to continuous development and improvement and the right is reserved to change products and services from time to time. City & Guilds cannot accept liability for loss or damage arising from the use of information in this publication.

## **City & Guilds**

**1 Giltspur Street**

**London EC1A 9DD**

**T +44 (0)844 543 0000 (Centres)**

**T +44 (0)844 543 0033 (Learners)**

**[www.cityandguilds.com](http://www.cityandguilds.com)**

**[learnersupport@cityandguilds.com](mailto:learnersupport@cityandguilds.com)**

# Contents

## Level 3 Software design fundamentals (7540-033/7630-348)

Assignment C

Introduction – Information for Candidates	2
Candidate instructions	3
Appendix A	7

# Level 3 Software design fundamentals (7540-033/7630-348)

## Assignment C

### Introduction – Information for Candidates

#### About this document

This assignment comprises all of the assessment for Software design fundamentals (7540-033/7630-348).

---

#### Health and safety

You are asked to consider the importance of safe working practices at all times.

You are responsible for maintaining the safety of others as well as your own. Anyone behaving in an unsafe fashion will be stopped and a suitable warning given. You will **not** be allowed to continue with an assignment if you compromise any of the Health and Safety requirements. This may seem rather strict but, apart from the potentially unpleasant consequences, you must acquire the habits required for the workplace.

#### Time allowance

The recommended time allowance for this assignment is **3 hours**.

# Level 3 Software design fundamentals (7540-033/7630-348)

## Candidate instructions

**Time allowance:** 3 hours

### Assignment set up:

This assignment is made up of **one** task

**Task A** – interpret documentation and design software components from a given specification

### Scenario

You work as a software developer for Complex Solutions. You have been asked to work as a team member to help design a prototype for a new compiler that the company are developing. The compiler is to compile the source code from a new programming language and convert the source code so that the software can be run via the Internet on any computer. The specification for the compiler is in Appendix A.

### Task A

In this task you are required to design part of the software for the prototype compiler. The following Event/Action chart has been provided by your team leader.

Event/Action chart			
Event/function/ procedure	Parameters	Action	Function/ procedure calls
private void mnuOpen		Opens the source code file and reads and displays the data in the editor window	
private void mnuSave		Saves the source code file	
private void mnuSaveAs		Saves the source code file with the entered filename	
private void mnuClose		Prompts to save source code file if it has changed and then closes the file	

<b>private void mnuExit</b>		<b>Prompts to save source code file if it has changed and then exits the software</b>	
private void mnuCompile		Compiles the source code	Init InitResWords ReadLines CheckSyntax
private void Init		Initialises arrays	
private void InitResWords		Sets up the reserved words in the symbol table	
private void ReadLines		Reads the lines in the source code file	GetWord CheckValidIdent HashSymbol
private boolean GetWord	by value: integer LineCounter - contains source code line number  by reference: char array Line – the input source code line boolean Comment – contains true if a comment spans more than one line otherwise false char array Word – the word found in a line	Finds a word or string in the passed Line parameter Removes comment text Replaces a comma with a space Outputs error message 003 or 004 if an error found Returns in the Comment parameter true if a comment spans more than one line or false Returns in the Word parameter the word or string found or an empty string Returns in the Line parameter the input line with the word, string or comment removed or an empty string Returns a boolean value true if no error found, false if an error found	
private boolean CheckValidIdent	by value: integer LineCounter – contains source code line number char array Word – holds the identifier to be checked	Checks that the Word parameter passed is a valid identifier ie contains valid letters a..z or A..Z or digits 0..9 but does not have a digit in the first position Outputs error message 005 if an error found Returns a boolean value true if no error found, false if an error found	
private void HashSymbol	by value char array InWord – holds the identifier to be hashed into the symbol table	Hashes the identifier into the symbol table and inserts data into the lexical records	CalcHash

	integer LineCounter - contains source code line number		
private integer CalcHash	by value WordIn	Calculates the hash value for the data in WordIn and returns the hashed value	
private void CheckSynTax		Works through the symbol table checking the syntax of the records against the syntax list for each instruction	
private void mnuRun		Runs the program using the data in the lexical records	

**1 Produce the design language algorithms for the following functions/procedures:**

- **mnuClose**
- **CheckValidIdent**
- **GetWord.**

2 Make sure that error messages as specified in the specification are output if an error occurs.

3 Make sure that the design follows the criteria listed below:

- the design conforms to the specification
- the program design language clearly shows
  - variable names and data types,
  - argument/parameter names and data types,
  - return value data types
- the design is consistent and complete
- quality criteria are met by the design.

4 Please use the Answer Sheet provided to complete Task A4.

Give a brief description of the role of software design in the IT Systems Development Life Cycle (SDLC).

5 Please use the Answer Sheet provided to complete Task A5.

Explain the purpose of encapsulation.

6 Please use the Answer Sheet provided to complete Task A6.

Explain which type of software is used to support the whole SDLC.

7 Please use the Answer Sheet provided to complete Task A7.

Explain **two** advantages of using pre-written library routines.

8 Please use the Answer Sheet provided to complete Task A8.

Describe **two** types of validation check and give an example for each.

**Note**

- Candidates should produce the following for their assessor:
  - program design language algorithms for the functions/procedures mnuClose, CheckValidIdent and GetWord

- Completed Answer Sheet.
- Ensure that your name is on all documentation
- If the assignment is taken over more than one period, all paperwork must be returned to the test supervisor at the end of each sitting.

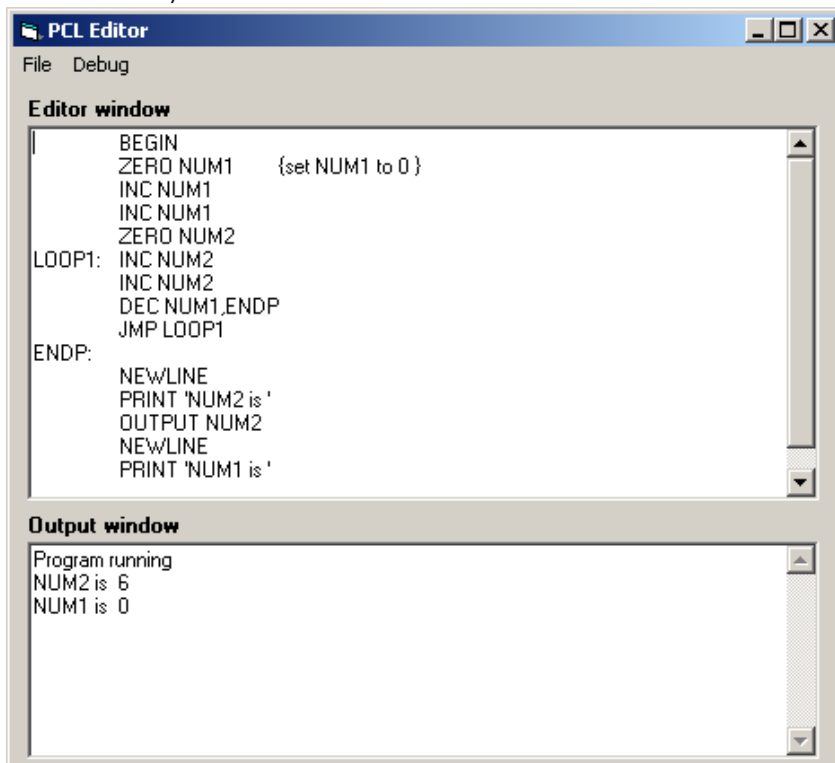


# Appendix A

## Specification

A compiler is required for a new programming language called Intermediate Computer Programming Language (ICPL) which is under development. The new programming language is intended for use on the Internet. The instructions are compiled into tokens and it is intended that the program can then be run on any computer.

The screen layout for the software is shown below.



The File menu contains the menu options - Open, Save, Save As, Close and Exit.

The Debug menu contains the menu options – Compile and Run.

The source code files that are input and output by the compiler have the extension **.cpl**

There is an Editor window and an Output window. The source code for a program can be entered into the Editor window and saved or an existing file can be read from disk.

When the Compile menu option is selected the source code is compiled and the compiler checks each line word by word. Comment lines are removed and any comma found is replaced by a space. For string data the start and end quote are checked for. If an error is found an error message is output. A lexical analysis is done to check that identifier names are valid. No entry is made in the symbol table or lexical record if an error is found. When the lexical analysis is complete, if no errors are found the compiler performs a syntax analysis. If any errors are found in the syntax of the code an error message is output.

The Run menu option will not run the program unless a compilation has been performed successfully first.

## Instruction set

The ICPL contains the following instruction set which is used to create a program.

### **ZERO** variablename

Sets a specified variable to 0 and continues to the next instruction.

### **INC** variablename

increases the specified variable by 1 and continues to the next instruction.

### **DEC** variablename,label

This is a conditional instruction. It checks the contents of the specified variable. If the contents of the specified variable are not zero then the contents of the specified variable are decreased by 1 and the next instruction is executed. If the specified variable contains zero then a jump is made to the specified label. This instruction is useful for executing and terminating a loop.

### **MOV** variablename, variablename

This instruction moves the value in the first specified variable to the second specified variable and then continues to the next instruction. This instruction is useful for saving values in a variable before executing a loop.

### **JMP** label

This instruction is an unconditional jump to a specified label in the program. This allows the order in which instructions are executed to be altered.

### **HALT**

This instruction signals the end of the program instructions.

{ text }

This format is used to insert comments into the program. Multiple line comments can be used.

### **INPUT** variablename

This instruction inputs a numeric integer in the range 0 to 500 into the specified variable and continues to the next instruction.

### **OUTPUT** variablename

This instruction outputs the contents of the specified variable and continues to the next instruction. This is useful to allow calculated results to be output.

### **PRINT** 'text'

This instruction outputs the text contained within the single quotes and then continues to the next instruction. It is used to make the output values meaningful.

### **NEWLINE**

This instruction outputs a newline and continues to the next instruction. It is required so that data is not output on a continuous line.

### **LABEL:**

This instruction is used for a label that can be branched to by the conditional and unconditional branch instructions (DEC and JMP).

## Error messages

Error messages are displayed in the Output window.

Error Code	Error message
001:	Cannot open file <i>filename</i>
002:	Cannot save file
003:	2 begin comment symbols, line no <i>linenumber</i>
004:	No endQUOTE found, line no <i>linenumber</i>
005:	Invalid identifier name <i>name</i> , line no <i>linenumber</i>
006:	<i>Line input</i> Syntax Error: Invalid label, line no <i>linenumber</i>
007:	<i>Line input</i> Syntax Error: Reserved word mis-spelt or missing, line no <i>linenumber</i>
008:	<i>Line input</i> Syntax Error: String expected, line no <i>linenumber</i>
009:	<i>Line input</i> Syntax Error: Identifier expected, line no <i>linenumber</i>
010:	<i>Line input</i> Syntax Error: Label name expected, line no <i>linenumber</i>
011:	No end comment symbol found
012:	Invalid data entered
013:	Must have successful compilation first
014:	Cannot run, file not open
015:	Syntax errors found: Compilation terminated
016:	Lexical Analysis errors found: Compilation terminated

Note that when the program is run one syntax error can cause several other syntax errors to be signalled. This is because the compiler becomes out of step with the start of an instruction and is a normal result for a syntax analysis. Also the line number may point to the line below where the actual error occurred.

## BNF Definition

<program>	::=	BEGIN<vsep><statementseq><vsep>END
<statementseq>	::=	<statement> <statement><vsep><statementseq>
<statement>	::=	<sep><stat> <sep><label>:<spaces><stat> <sep><label>:
<label>	::=	<letter> <label><letter> <label><digit>
<letter>	::=	A B C.....Y Z a b c.....y z
<digit>	::=	0 1 2 3 4 5 6 7 8 9
<sep>	::=	<spaces> <>null> <tab>
<>null>	::=	
<stat>	::=	<IOstat> <calcstat> <haltstat>
<haltstat>	::=	HALT
<IOstat>	::=	<INstat> <OUTVstat> <OUTPstat> <NEWLstat>
<calcstat>	::=	<zerostat> <incstat> <jmpstat> <movstat> <decstat>
<INstat>	::=	INPUT <spaces><varname>
<OUTVstat>	::=	OUTPUT <spaces><varname>
<OUTPstat>	::=	PRINT <spaces>'<text>'
<NEWLstat>	::=	NEWLINE
<zerostat>	::=	ZERO<spaces><varname>
<incstat>	::=	INC<spaces><varname>
<jmpstat>	::=	JMP<spaces><label>
<movstat>	::=	MOV<spaces><varname>,<varname>
<decstat>	::=	DEC<spaces><varname>,<label>
<varname>	::=	<letter> <varname><digit> <varname><letter>
<vsep>	::=	<CR> {<text>}<vsep> <CR><vsep> <spaces><vsep> where CR is an implementation of carriage return/linefeed
<text>	::=	<text><char> <char>
<char>	::=	any ASCII char except { } ' "
<spaces>	::=	b b<spaces> where b represents a space

The | symbol represents OR

The definition restricts labels and variables so that they cannot start with a digit.

## A sample program

BEGIN

{ filename TEST.CPL }

{The syntax in this program is correct as defined for the language}

{ A program to multiply 2 positive integers input into variables  
NUM1 and NUM2. The multiplication is done by using multiple additions.  
The value in NUM1 is added to RESULT for the number of times in NUM1 }

```
NEWLINE
PRINT 'This program will multiply 2 positive integers'
NEWLINE
NEWLINE
PRINT 'Input first number'
INPUT NUM1
NEWLINE
PRINT 'Input second number'
INPUT NUM2
ZERO RESULT      { set RESULT to 0 }
MOV NUM2,STORE
```

{Store the contents of the variable NUM2 in the variable STORE }

LOOP1: DEC NUM1,ENDP

DEC NUM2,ENDP { test if initial values = 0 }

LOOP2:

INC RESULT { Increase RESULT by 1 }

DEC NUM2,LOOP3 { If NUM2 = 0 then exit loop }

JMP LOOP2 { If NUM2 not = 0 repeat loop }

LOOP3:

MOV STORE,NUM2

{ Restore the value from STORE to NUM2 }

JMP LOOP1 { Jump to repeat loop1 }

ENDP:

NEWLINE

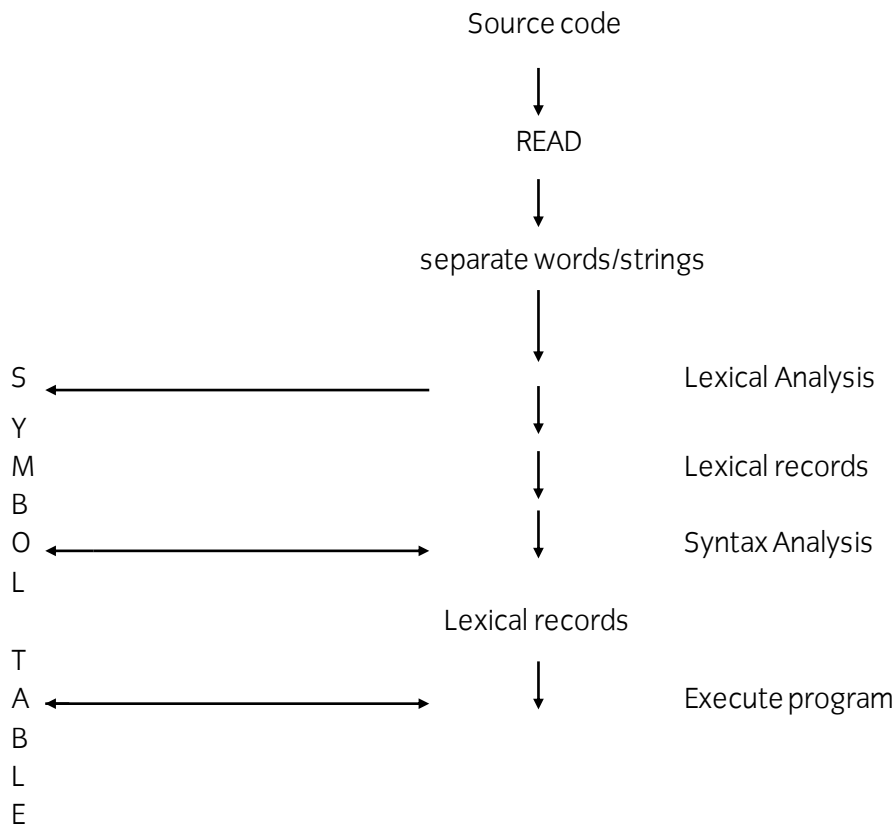
PRINT 'Result = '

OUTPUT RESULT { Output the result from RESULT }

NEWLINE

HALT

END



The source code file is read and for each line, comments are removed, a comma is replaced by a space and each word or string in the line is separated. The word in the line is checked to make sure that it is a valid identifier and then stored in the symbol table. The subscript for the string array is stored in the lexical record and the string is stored in the string array.

## Symbol table

Reserved words are hashed into the symbol table each time the compiler is run using the same algorithm as is used to search for and insert identifiers into the symbol table.

---

**Published by City & Guilds**  
**1 Giltspur Street**  
**London**  
**EC1A 9DD**  
**T +44 (0)844 543 0000 (Centres)**  
**T +44 (0)844 543 0033 (Learners)**  
**[www.cityandguilds.com](http://www.cityandguilds.com)**

**City & Guilds is a registered charity  
established to promote education  
and training**