-SQA-SCOTTISH QUALIFICATIONS AUTHORITY

NATIONAL CERTIFICATE MODULE: UNIT SPECIFICATION

GENERAL INFORMATION

-Module Number- **8110105** **-Session- 1995-96**

-Superclass- **CB**

-Title- **DEVELOPING SOFTWARE 1**

-DESCRIPTION-

GENERAL COMPETENCE FOR UNIT: Appreciating the features of contemporary software and understanding how software is produced.

OUTCOMES

1. explore the features of existing software;

2. describe the stages in developing new software;

3. investigate contemporary methods of producing software.

CREDIT VALUE: 1 NC Credit

ACCESS STATEMENT: No previous qualifications or experience to access this unit.

For further information contact: Committee and Administration Unit,, SQA, Hanover House, 24 Douglas Street, Glasgow G2 7NQ.

Additional copies of this unit may be purchased from SQA (Sales and Despatch section). At the time of publication, the cost is £1.50 (minimum order £5.00).

NATIONAL CERTIFICATE MODULE; UNIT SPECIFICATION

STATEMENT OF STANDARDS

UNIT NUMBER: 8110105

UNIT TITLE: DEVELOPING SOFTWARE 1

Acceptable performance in this unit will be the satisfactory achievement of the standards set out in this part of the specification. All sections of the statement of standards are mandatory and cannot be altered without reference to SQA.

OUTCOME

1. EXPLORE THE FEATURES OF EXISTING SOFTWARE

PERFORMANCE CRITERIA

(a) Exploration is effective in identifying the features of the software.
(b) Use of documentation is efficient and effective.
(c) Identification of the features of each type of program is accurate.
(d) Identification of the characteristics of high quality programs is accurate.
(e) Identification of the characteristics of high quality documentation is accurate.

RANGE STATEMENT

Features: user interface; functionality; speed; robustness; documentation (including on-line help).

Program types: systems; applications (including leisure and business).

EVIDENCE REQUIREMENTS

Evidence of actual performance to show that the candidate can explore software as detailed in Performance Criteria (a) and (b) for at least three software products.

Written or oral evidence is needed to show that the candidate can identify the features as detailed in Performance Criteria (c) to (e) for at least three software products.

OUTCOME

2. DESCRIBE THE STAGES IN DEVELOPING NEW SOFTWARE

PERFORMANCE CRITERIA

(a) Description includes all stages.
(b) Description of each stage is accurate.
(c) Identification of the sequence of each stage is correct.
(d) Description of the iterative nature of software development is accurate.

RANGE STATEMENT

Stages: analysis; design; implementation; testing; documentation; review.

EVIDENCE REQUIREMENTS

Written or oral evidence is needed to show the candidate can describe the stages in developing software as detailed in Performance Criteria (a) to (d).

OUTCOME

3. INVESTIGATE CONTEMPORARY METHODS OF PRODUCING SOFTWARE

PERFORMANCE CRITERIA

(a) Conduct of investigation is efficient and effective.
(b) Description of types of contemporary programming languages is accurate.
(c) Description of role of members of development team is accurate.

RANGE STATEMENT

Programming languages: procedural; event driven; object orientated; functional.

Roles: project leader; systems analyst; programmer.

EVIDENCE REQUIREMENTS

Evidence of actual performance to show that the candidate can investigate contemporary means of producing software as detailed in Performance Criterion (a).

Written or oral evidence of the candidate's knowledge and understanding of contemporary means of producing software as detailed in Performance Criteria (b) and (c).

ASSESSMENT

In order to achieve this unit, candidates are required to present sufficient evidence that they have met all the performance criteria for each outcome within the range specified. Details of these requirements are given for each outcome. The assessment instruments used should follow the general guidance offered by the SQA assessment model and an integrative approach to assessment is encouraged. (See references at the end of support notes).

Accurate records should be made of the assessment instruments used showing how evidence is generated for each outcome and giving marking schemes and/or checklists, etc. Records of candidates' achievements should be kept. These records will be available for external verification.

SPECIAL NEEDS

In certain cases, modified outcomes and range statements can be proposed for certification. See references at end of support notes.

NATIONAL CERTIFICATE MODULE: UNIT SPECIFICATION

SUPPORT NOTES

UNIT NUMBER: 8110105

UNIT TITLE: DEVELOPING SOFTWARE 1

SUPPORT NOTES: This part of the unit specification is offered as guidance. None of the sections of the support notes is mandatory.

NOTIONAL DESIGN LENGTH: SQA allocates a notional design length to a unit on the basis of time estimated for achievement of the stated standards by a candidate whose starting point is as described in the access statement. The notional design length for this unit is 40 hours. The use of notional design length for programme design and timetabling is advisory only.

PURPOSE This module may be taken as a free-standing module in a wide variety of programmes. It is suitable for candidates undertaking a wide range of National Certificate awards.

SQA publishes summaries of NC units for easy reference, publicity purposes, centre handbooks, etc. The summary statement for this unit is as follows

This unit has been designed to serve as a simple introduction to computer programming for students who are interested in producing computer software or who wish to appreciate the software development process. Students will explore the features of existing software (such as games programs and commercial software) and will be introduced to the stages in producing new software together with contemporary methods of producing it. This module does not require students to write computer programs (although students may be asked to do this as part of the learning process); subsequent modules will focus on programming.

CONTENT/CONTEXT Corresponding to Outcomes 1-3:

1. The aim of this outcome is to expose students to a wide range of pre-written software to expose them to good (and bad) examples of computer programs and associated documentation. It is anticipated that the majority of students' time will be spent on this outcome. Centres are encouraged to provide students with the opportunity to experience a range of diverse programs. A minimum of three items of software must be explored. This might include a small utility program (such as a program to defragment disks), a business application (such as a spreadsheet program) and leisure software (such as a computer game). Diversity within a class of software should also be explored. For example, games software includes adventure games, 3D combat games and simulations. Students should be

exposed to as wide a range of software as time permits. This outcome should prepare students for subsequent outcomes by providing them with experience of a variety of contemporary software. In particular, the strengths and weaknesses of each program should be explored. For example, a specific program might have a poor user interface; another program might run slowly; another may have poor game play. There is scope for group discussion of the pros and cons of specific programs. An alternative approach might involve students in writing personal reviews of specific programs. Students should also be encouraged to critically appraise the documentation that comes with the program.

2. This outcome focuses on the software development process. It is not required that the student gains an in-depth understanding of any one stage; rather; s/he should be exposed to typical activities. For example, students should gain an appreciation of the review stage together with an appreciation of how this is carried out. A crucial point to reinforce is the iterative nature of the software development cycle. While this outcome is knowledge-based, it can be made more interesting through the use of video, case studies and guest lecturers. For example, a local practitioner could be invited to the centre to talk to students about the projects in which s/he has been involved. Case studies should relate to failed software projects as well as successful ones.

3. This outcome provides an opportunity to investigate the methods of producing software. It focuses on the people involved in the software development process and the range of tools they have available. Given the introductory nature of this unit, it is not required that students gain a detailed knowledge of any specific role or software tool. For example, lecturer exposition of the types of programming language should be light but sufficient to reinforce the distinctions between major classes of programming language. Important instances within a class should also be introduced such a C and Visual basic. The description of the role of each member within the development team need not be too detailed; it is sufficient to outline the major tasks of each person. The importance of team work should be stressed. This outcome is descriptive and can be enlivened through the use of video, case studies, guest speakers and a visit to a software company. For example, there have been various television programmes on the developments of high profile software projects. Students should be encouraged to conduct their own research through reading, questioning and participating in some of the suggested activities.

APPROACHES TO GENERATING EVIDENCE A candidate-centred, resource-based learning approach is recommended. During the work of the module, candidates should have several opportunities to develop their practical skills and should be assessed at appropriate points. Terminology should be presented in context throughout the module. Where the candidate is unsuccessful in achieving an outcome, provision should be made for remediation and re-assessment.

The evidence of competence for this module should be generated naturally during the life of the module. It is recommended that the focus of the module is Outcome 1 with the other outcomes introduced at appropriate points within the module. For example, the roles of each person within a project team (Outcome 3) could be introduced when this subject arises naturally in discussing a specific program as part of Outcome 1.

ASSESSMENT PROCEDURES Centres may use instruments of assessment which are considered to be most appropriate. Examples of instruments of assessment which could be used are as follows:

Outcome 1

Assignment requiring the student to explore and evaluate a wide range of software. This assignment might incorporate the cognitive and practical aspects of the evidence requirements. For example, s/he may be observed running software and completing a pro-forma identifying the characteristics of each program.

Outcome 2

Extended response question requiring the candidate to describe the main stages involved in the software development process; the extent of response could be between 500-750 words.

Outcome 3

Assignment involving the candidate in researching the work of software development teams and the nature of contemporary programming languages and producing a short summary of their findings. The extent of response could be between 750-1000 words.

There is an opportunity to integrate the assessment for Outcomes 2 and 3 into a single assessment activity requiring the candidate to describe the stages, roles and tools involved in software production.

The required performance evidence for each of the outcomes could be combined into a single observation checklist which could be completed throughout the life of the unit.

PROGRESSION This module contributes towards the National Certificate (level II) Information Technology and is the first in a series of modules entitled ‘Developing Software’. Students may progress to the second level module entitled ‘Developing Software 2’ which involves writing computer programs.

RECOGNITION Many SQA NC units are recognised for entry/recruitment purposes. For up-to-date information see the SQA guide ‘Recognised Groupings of National Certificate Modules’.

REFERENCES

1. Guide to unit writing.
2. For a fuller discussion on assessment issues, please refer to SQA's Guide to Assessment.
3. Procedures for special needs statements are set out in SQA's guide 'Students with Special Needs'.
4. Information for centres on SQA's operating procedures is contained in SQA's Guide to Procedures.
5. For details of other SQA publications, please consult SQA's publications list.