Version 1.1

AQA General Certificate of Secondary Education

4512

GCSE in Computer Science

Component 1: Practical programming
CONTROLLED ASSESSMENT 25 HOURS

Suggested time allocation

- **Analysis 3 hours** (approx 4 lessons)
- **Design 4 hours** (approx 5 lessons)
- **Implementation 9 hours** (approx 10.5 lessons)
- **Testing 6 hours** (approx 6.5 lessons)
- **Evaluation/ Improvements 3 hours** (approx 4 lessons)

TOTAL: 25 hours (approx 30 lessons)

NB it is permissible to extend the recommended hours to 28!

St John Fisher Catholic High School

CONTROLLED ASSESSMENT GCSE COMPUTING AQA

The CONTROLLED TASK

Read through your examination booklet SEVERAL TIMES to make sure THAT you understand ALL the REQUIREMENTS fully and making any necessary notes
CREATE a NEW FOLDER in your exam area labelled

GCSE 2014 CONTROLLED ASSESSMENT ONE

Create 5 SUB FOLDERS

(1) analysis
(2) design
(3) implementation
(4) test plan & testing
(5) evaluation & suggested improvements

Using the scenario from your examination paper COMPLETE the five tasks listed below

(1) Analyse the problem making a note of the key points and facts, make sure that you are certain what the inputs and outputs are and where the data comes from. You must draw up a list of project objectives.
HINT a good analysis will give details of

- The Client and any other End Users
- The Background to the problem
- The facts you have been given by your client
- ALL Outputs, Inputs, processes, and Storage
- A schedule
- A list of DISCRETE & MEASURABLE OBJECTIVES, that is things you hope to achieve that you can demonstrate using screen shots.
- It may also include some additional (background) research

(2) Draw up a design using flow charts, algorithms drawings and text as appropriate

HINT a good DESIGN will include

- A summary of the analysis
- A systems overview
- An IPSO for the system you are designing
- Algorithms for every process
- Flowcharts for every process
- Form and screen designs
- A data dictionary showing ALL variables and Objects, their type, their purpose
- Any validation you will put in to prevent DEEs
- Any colour coding, user messages etc which you intend to use to make the product user friendly
- Any password protection
- Data flow diagrams

REMEMBER: your design should be such that it could enable another programmer to create your product

(3) Create a programmed solution ensuring that you submit a copy of your fully annotated coding

To DOCUMENT your solution you will need to

- SCREEN shot DESIGN and ENDUSER views of your forms, to demonstrate that the product works
- Annotate these screen shots, and x-reference to the code you have used
- Show a complete listing from Visual Basic of ALL the code that you have written
- Every line of code must be annotated and where possible x-referenced to screen shots

(4) Thoroughly test your solution using a plan cross retrenched to numbered screen shots

A good TESTING SECTION will have

- A TEST PLAN with
- Test number (x-referenced to screen shots), test description, reason for test, expected result, (a column which will be blank until the tests have been done for TEST RESULTS), a comment on the result and what action if any is required.
- TESTING SCREEN SHOTS (x-referenced to test plan) showing the product being tested with VALID (typical and extreme data) and INVALID data
- Structured FEEDBACK from another person on the effectiveness of the product.

(5) Thoroughly evaluate your solution in terms of (a) how it meets your project objectives (b) how elegant & efficient the solution is and how well you have demonstrated your use and understanding of code (3) how user friendly it is
Make suggestions for improvement/ development
Suggest how these might be achieved

A good evaluation will have

- Comments on how each project objective was met
- How effectively you worked (e.g. good use of code, through planning etc.)
- How well you have documented your work: could another person follow it?
- How user friendly is it?
- Is it suited to your client?
- Feedback from others, using questionnaires you have issued
- Suggested improvements
- Methods of implementing these improvements
- An attempt at implementing some of the improvements

CONTROLLED ASSESSMENT GCSE COMPUTING AQA

The portfolio for each controlled assessment should be divided into four sections:

1. **Design of the Solution**
2. **Solution Development**
3. **Programming Techniques Used**
4. **Testing and Evaluation**

Overview

Design of solution 9 marks
Solution development 9 marks
Programming techniques used 36 marks
Testing and evaluation 9 marks

Total 63 marks

Detailed breakdown

Design of solution (0–9 marks available)

The design of solution should include:
1. An explanation of what the problem is and what the solution should be capable of in terms of the needs of the user and their stated objectives
2. A plan showing the high level overview of how the solution will be constructed (using any suitable method)
3. explained/annotated pseudocode (or suitable alternative) showing the main blocks within the proposed solution.

Solution development (0–9 marks available)

The solution development should include:
1. Evidence of the final solution in the form of annotated code
2. Evidence that the final solution meets the original needs of the user.

Programming techniques used (0–36 marks available)

The programming techniques used should include:
1. Annotated evidence of the different programming techniques used
2. Annotated evidence showing how the different programming techniques used combine to form a complete solution that solves the original problem
3. Annotated evidence showing that the solution has been coded efficiently
4. A discussion of any data structures created with an explanation of why they are required
5. A solution that has been made robust by using techniques such as input validation, assertions and error handling, as appropriate to the language used.

Testing and evaluation (0–9 marks available)

The testing and evaluation should include:
1. A full test plan covering all of the major success criteria for the problem
2. Evidence that the tests have been carried out with the results being documented
3. Any remedial action that has been taken as a result of testing
4. An evaluation of how the solution meets the original needs of the user
The quality of the written communication will be assessed in this section.

Sample IPSO (from GARDEN PROJECT)

INPUTS	OUTPUTS
Length and width of the lawn Length and width of the concrete patio Length and width of the wooden deck Length and width of the rectangular pond Number of water features Number of garden lights	Length and width of the lawn area Total area of the lawn (m^2) Cost of the lawn (per m^2) Total cost of the lawn Length and width of the concrete patio area Total area of the concrete patio (m^2) Cost of the concrete patio (per m^2) Total cost of the concrete patio Length and width of the wooden deck area Total area of the wooden deck (m^2) Cost of the wooden deck (per m^2) Total cost of the wooden deck Length and width of the rectangular pond area Total area of the rectangular pond (m^2) Cost of the rectangular pond (per m^2) Total cost of the rectangular pond area Number of water features Cost of one water feature Total cost of the water features Number of garden lights Cost of one garden light Total cost of the garden lights Total time to complete the work (labour) in hours Cost of the work (per hour) Total cost of the work (labour) This is calculated by adding together the total costs from items 1 to 7 above. Displayed/printed report
PROCESSING	**STORAGE**
CALCULATION of Total area of the lawn (m^2) Cost of the lawn (per m^2 Total cost of the lawn Total area of the concrete patio (m^2) Cost of the concrete patio (per m^2) Total cost of the concrete patio Total area of the wooden deck (m^2) Cost of the wooden deck (per m^2) Total cost of the wooden deck Total area of the rectangular pond (m^2) Cost of the rectangular pond (per m^2) Total cost of the rectangular pond area Total cost of the water features Number of garden lights Total cost of the garden lights Total time to complete the work (labour) in hours Cost of the work (per hour) Total cost of the work (labour) Retrieve costs form text file Retrieve customer quote CREATE a monthly report that compares the total cost of all materials purchased per month across all jobs undertaken.	Use of variables to be assigned to all data entered and to All data calculated Store final quote STORE Prices /costs of raw material in a text file STORE total cost of all materials purchased per month across all jobs undertaken

SAMPLE ALGORITHMS in pseudocode FROM GARDEN PROJECT
(all the processes will need algorithms)

CALCULATION of costs (materials)

```
START
'LAWN
        Var CostOfLawn as double
        Input WidthL,LengthL as double
        Input CostL as double
        CostOfLawn=widthL*LengthL*CostL

'PATIO
        Var CostOfPatio as double
        Input WidthP, lengthP as double
        Input CostP as double
        CostOfPatio=WidthP*LengthP*CostP

'DECK
        Var CostOfDeck as double
        Input WidthD, lengthD as double
        Input CostD as double
        CostOfDeck=WidthD*LengthD*CostD

'POND
        Var CostOfPond as double
        Input WidthPond, lengthPond as double
        Input CostPond as double
        CostOfPond=WidthPond*LengthPond*CostPond

'WATER FEATURE
        Var CostOfWaterFeature as double
        Input QTYofWaterFeature as integer
        Input CostPerFeature as double
        CostOfWaterFeature=QTYofWaterFeature*CostPerFeature

'GARDEN LIGHTS

        Var CostOfGardenLights as double
        Input QTYofGardenLight as integer
        Input CostPerGardenLight as double
        CostOfGardenLight=QTYofGardenLight*CostPerGardenLight

'TOTALS
Var TOTALMATERIALS as double

TOTALMATERIALS= CostOfGardenLight+ CostOfWaterFeature+ CostOfPond+
CostOfDeck+ CostOfPatio+ CostOfLawn
END
```

ALGORITHMS

CALCULATION of costs (labour)

```
START
'LAWN
        Var LabourLHours, labourCostL as double
        Input WidthL,LengthL as double
        LabourLHours=(widthL*LengthL)/3
        LabourCostL=LabourLHours*16.49

'PATIO
        Var LabourPHours, LabourCostP as double
        Input WidthP,LengthP as double
        LabourPHours=(widthP*LengthP)/3
        LabourCostP=LabourPHours*16.49

'DECK
        Var LabourDHours, LabourCostD as double
        Input WidthD,LengthD as double
        LabourDHours=(widthD*LengthD)/2
        LabourCostD=LabourDHours*16.49

'POND
        Var LabourPondHours, LabourCostPond as double
        Input WidthPond,LengthPond as double
        LabourPondHours=(widthPond*LengthPond)*0.75
        LabourCostPond=LabourPondHours*16.49

'WATER FEATURE
        Var LabourWFHours, LabourCostWF as double
        Input WidthWF,LengthWF as double
        LabourWFHours=(widthWF*LengthWF)*1
        LabourWFHours*16.49


'GARDEN LIGHTS

        Var LabourGLHours, LabourCostGL as double
        Input WidthGL,LengthGL as double
        LabourGLHours=(widthGL*LengthGL)/6
        LabourCostGL=LabourGLHours*16.49

'TOTALLABOUR
VAR TotalLabour as double
TotalLabour=LabourCostG+ LabourWFHours+ LabourCostPond+ LabourCostD+
LabourCostP+ LabourCostL

END
```

ALGORITHMS

CALCULATION of costs (ALL)

START

VAR TotalCostCustomer, as double

TotalCostCustomer= TotalLabour+ TOTALMATERIALS

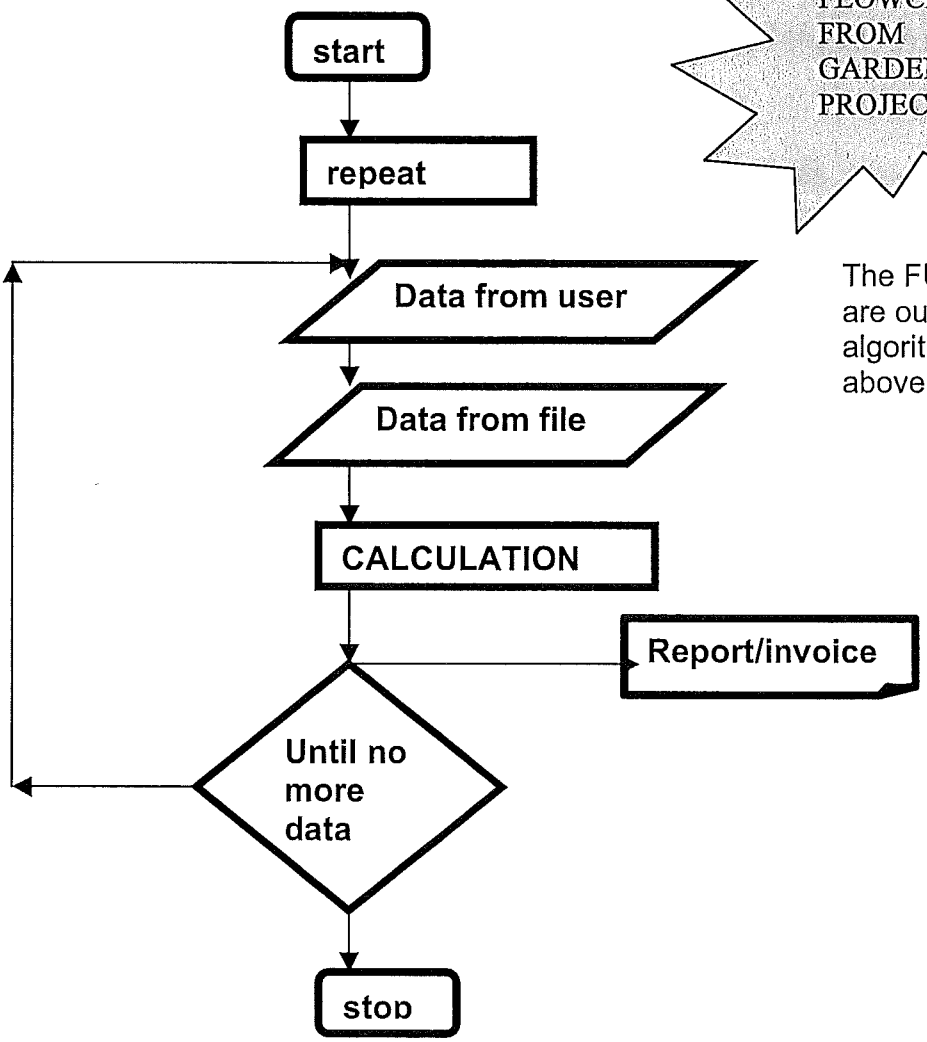
ALGORITHMS
START
'STORAGE of Data
Var COSTS as text file
Var
CostOfLawn,CostOfpatio,CostofDeck,CostOfPond,CostOfWaterFeature,CostofGardenLights as double
Var LabourCostG, LabourWFHours, LabourCostPond, LabourCostD, LabourCostP, LabourCostL
OPEN COSTS
REPEAT
Write
CostOfLawn,CostOfpatio,CostofDeck,CostOfPond,CostOfWaterFeature,CostofGardenLights LabourCostG, LabourWFHours, LabourCostPond, LabourCostD, LabourCostP, LabourCostL to file
Until no more data
CLOSE File

END

ALGORITHMS
START
'RETRIEVAL of Data
Var COSTS as text file
Var
CostOfLawn,CostOfpatio,CostofDeck,CostOfPond,CostOfWaterFeature,CostofGardenLights as double
Var LabourCostG, LabourWFHours, LabourCostPond, LabourCostD, LabourCostP, LabourCostL
OPEN COSTS
REPEAT
READ
CostOfLawn,CostOfpatio,CostofDeck,CostOfPond,CostOfWaterFeature,CostofGardenLights LabourCostG, LabourWFHours, LabourCostPond, LabourCostD, LabourCostP, LabourCostL from file
Until no more data
CLOSE File

END

INVOICE FLOWCHART
Showing Looping



SAMPLE FLOWCHART FROM GARDEN PROJECT

The FULL procedures are outlined in the algorithms listed above

FLOW CHARTS SHOW THAT YOU HAVE USED COMPLEX PROGRAMMING STRUCTURES

This is one of the things that the examiner will look for

You will need to show

- **INITIALISATION**

Setting the variables to values before you start, usually) or 1

- **LOOPING**

Allowing parts of the program to loop back on itself e.g. entering 7 numbers one after the other, or reading data from a text file

- **REPETITION(ITERATION)**

Allowing the user to repeat data entry or restart for example

- **SELECTION**

Choosing from different options using IF STATEMENTS for example

- **FUNCTIONS**

Writing sub routines to do calculations and other tasks e.g. validation file reading etc

SAMPLE DATA DICTIONARY FROM GARDEN EXAMPLE

A good way of showing the examiner that you have though abut your design and also a good way of helping you as you work through your code is to create a DATA DICTIONARY

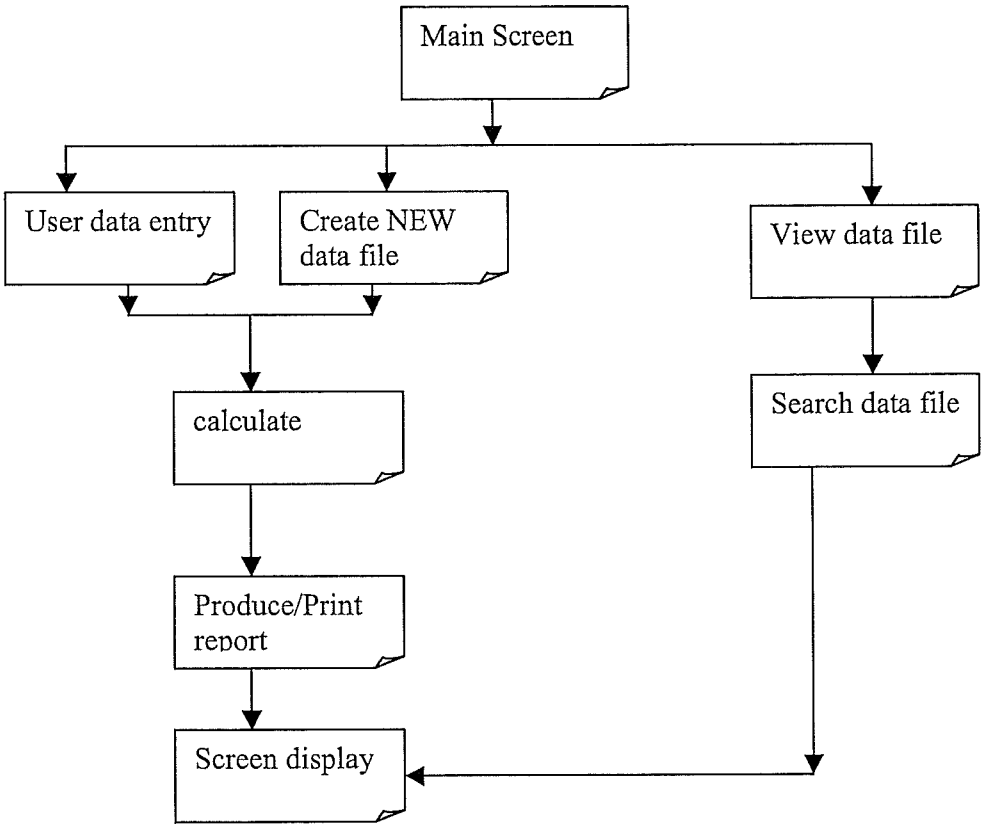
A data dictionary is a data table which describes all the data and objects in your project.

A section of a sample data dictionary is shown below

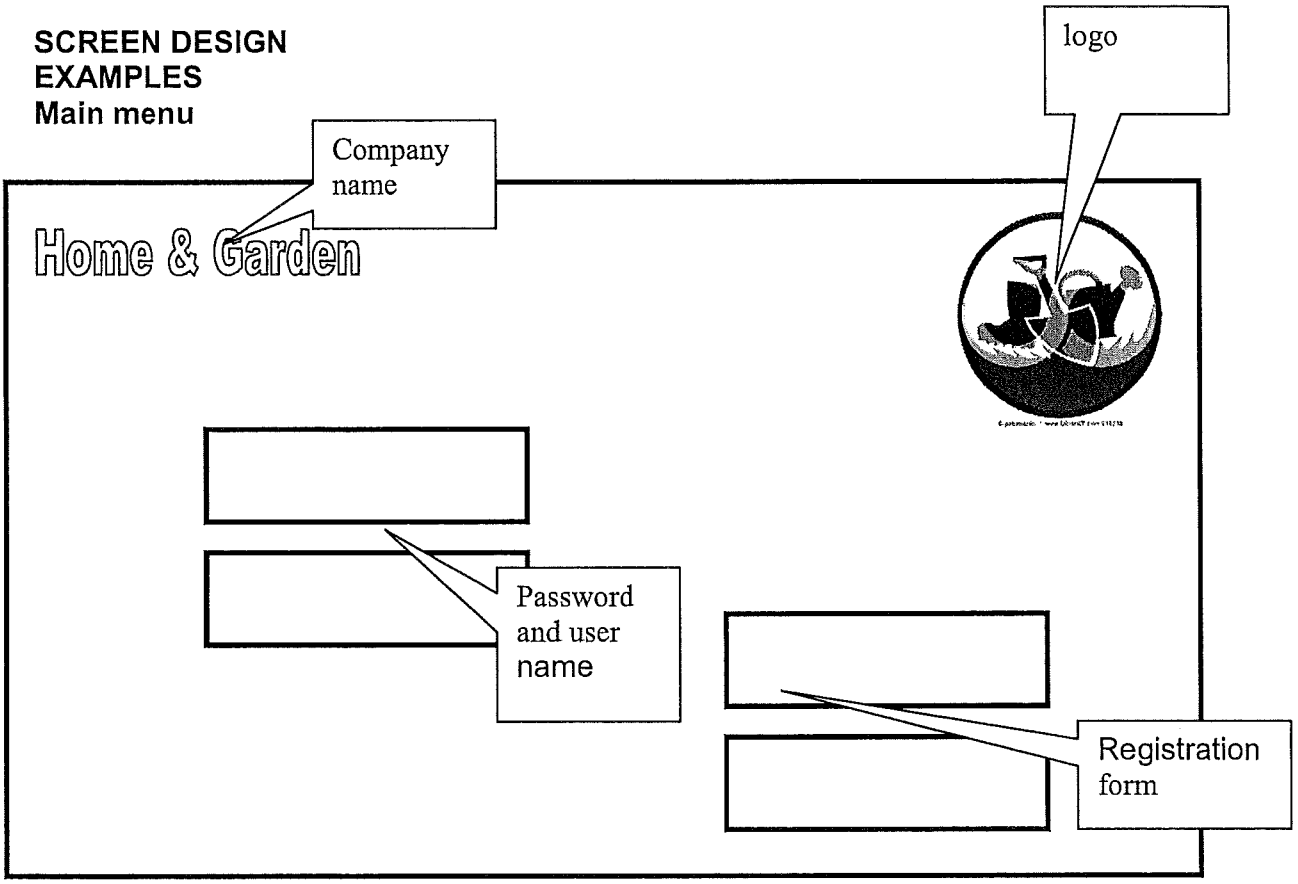
NAME	TYPE	USE
TxtStudentName	Object/text box	User enters student name into text box
LblStudentName	Object/label	Displays the words STUDENT NAME over the empty text box (TxtStudentName)
StudentName	Variable/string	Assigns data entered by user via TxtStudentName to StudentName
TxtStudentDOB	Object/text box	User enters student date of birth into text box
LblStudentDOB	Object/label	Displays the words "STUDENT Date of Birth" over the empty text box (TxtStudentDOB)
StudentDOB	Variable/date time	Assigns data entered by user via TxtStudentDOB to StudentDOB

USER INTERFACE EXAMPLE

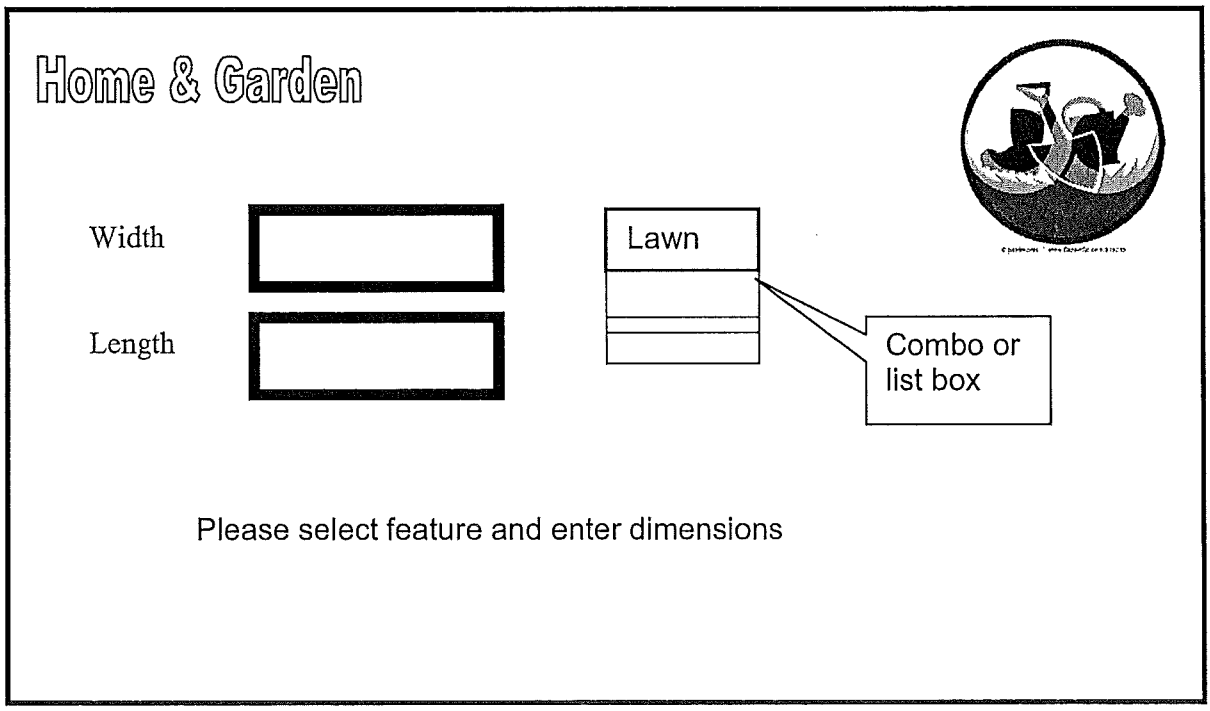
Form design: NAVIGATION



SCREEN DESIGN EXAMPLES
Main menu



User Data Input



EXAMNPLE
OBJECTIVES
FROM GARDEN
TASK:

Create A System To:

- Input and store customer order data, e.g. required dimensions of lawn, deck etc
- Input and Store costs and pricing data in text file
- CALCULATE Total cost of the lawn
- CALCULATE Total cost of the concrete patio
- CALCULATE Total cost of the wooden deck
- CALCULATE Total cost of the rectangular pond area
- CALCULATE Total cost of the water
- CALCULATE Total cost of the garden lights
- CALCULATE Total cost of project
- CREATE CUSTOMER QUOTE from outputs above
- Save CUSTOMER QUOTE as separate file
- Retrieve standard costs from text file
- Retrieve customer quote from dat. or txt. file
- CREATE a monthly report that compares the total cost of all materials purchased per month across all jobs undertaken
- Save report
- Display/print report/invoice/quote
- Provide a user friendly GUI which will help eliminate data entry errors and prove easy (intuitive) for the client to use
- Provide a USER MANUAL for a novice user
- Provide full systems documentation to allow for efficient systems maintenance

The SYSTEM CREATED will be measured against these original objectives!