

**Computer and
Information Science
University/College
Preparation (ICS3M)**

Ergonomic Risk Survey

The Task

Students were presented with the following scenario and instructions:

Healthy Valley Medical Centre uses computers in many different environments, such as offices and patient treatment areas. Healthy Valley is concerned with determining if their employees are using their computers in an ergonomically correct fashion. You are working as a co-op student and have been asked to create a 5–7 question ergonomic risk survey that will be implemented as a computer program. The program must provide an ergonomic risk level and suggested improvements for each workstation. The data will be collected in a common file for management use.

Final Product

Each student was to submit:

- a draft of the survey including a scoring guide and suggested improvements for management approval;
- a program plan in the form of pseudocode, flowchart, or diagram;
- a hard copy of the program including documentation and a program test report.

Expectations Addressed in the Exemplar Task

This task gave students the opportunity to demonstrate achievement of all or part of each of the following selected expectations from two strands: Skills and Processes, and Impact and Consequences.

Students will:

1. use selection structures, counted and conditional loops, and nested selection and loop structures;
2. develop appropriate algorithms in text or diagram form to solve problems and verify solutions;
3. use appropriate strategies to avoid potential health and safety problems associated with computer use, such as musculoskeletal disorders and eye strain;
4. incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;
5. write programs that access sequential files;
6. test completed programs with a full range of valid data to ensure that all components work as expected.

For information on the process used to prepare students for the task and on the materials and resources required, see the Teacher Package, reproduced on pages 105–115 of this document.

Task Rubric – Ergonomic Risk Survey

Expectations*	Criteria	Level 1	Level 2	Level 3	Level 4
Knowledge/Understanding					
The student:					
1	– demonstrates understanding of efficient and effective use of fundamental programming constructs	– demonstrates limited understanding of fundamental programming constructs	– demonstrates some understanding of fundamental programming constructs	– demonstrates considerable understanding of fundamental programming constructs	– demonstrates thorough understanding of fundamental programming constructs
Thinking/Inquiry					
The student:					
2	– develops the program plan effectively, using pseudocode, flowchart, or diagram	– develops the program plan with limited effectiveness	– develops the program plan with some effectiveness	– develops the program plan with considerable effectiveness	– develops the program plan with a high degree of effectiveness
3	– determines effective ergonomic improvements for each workstation	– determines health and safety improvements that are effective in a limited way	– determines somewhat effective health and safety improvements	– determines effective health and safety improvements	– determines highly effective health and safety improvements
Communication					
The student:					
4	– provides accurate and effective internal and external documentation according to given standards	– provides internal and external documentation that is limited in accuracy and effectiveness	– provides somewhat accurate and effective internal and external documentation	– provides internal and external documentation that is accurate and effective to a considerable degree	– provides highly accurate and effective internal and external documentation
Application					
The student:					
5	– successfully accesses a master survey data file	– accesses a master survey data file with limited success	– accesses a master survey data file with some success	– accesses a master survey data file with considerable success	– accesses a master survey data file with a high degree of success
6	– effectively tests the completed program with a full range of possible responses	– tests the program with limited effectiveness	– tests the program with some effectiveness	– tests the program with considerable effectiveness	– tests the program with a high degree of effectiveness

*The expectations that correspond to the numbers given in this chart are listed on page 8.

Note: A student whose overall achievement at the end of a course is below level 1 (that is, below 50%) will not obtain a credit for the course.

Notes on Programming Languages

The student samples have been written in various programming languages. The following notes identify the language used in particular samples and provide details of each language, for readers' information. Words that appear in italics represent actual code words in the language.

Visual Basic – Low Level 1

Visual Basic is an event-driven programming language. Visual Basic programs are composed of subroutines that are called by events. The following notes pertain to the Visual Basic programming language:

- the prefix *cmd* indicates a command button object, the prefix *opt* indicates an option button object, and the prefix *lbl* indicates a label object;
- each event is blocked off by a descriptive header and closing statement. For example, *Private Sub cmdnext_Click() ... End Sub* defines the piece of code that will be executed when the user generates a click-event by clicking the mouse on the command button named *cmdnext*. It is common to indent the code inside the event subroutine (not demonstrated in the Low Level 1 sample);
- the *Dim* statement is used to declare variables local to the form or event (e.g., in the Low Level 1 program, the student has declared *question* local to the form and *strname* local to the *cmdnext_Click()* event);
- the *MsgBox* statement generates a message pop-up with an OK button;

- the dot operator, a period (*.*), is used to access or modify the properties of an object. The statement *opt1.Value* returns the truth value of the *.Value* property of the *opt1* option button object. The statement *lblscore.Caption = intscore* sets the *.Caption* property of the *lblscore* label object equal to *intscore*;
- the *.Hide* and *.Show* commands are used to change focus between different form windows in a program (e.g., in the Low Level 1 sample, the student has written the program using multiple form windows and uses *.Hide* and *.Show* to switch between them);
- the *Unload* command removes a form from memory (e.g., in the Low Level 1 sample, the student has removed only *Form5*);
- *Open*, *Write*, and *Close* are used to control the contents of a file. The *For Append* ensures that information is written to the end of the file, instead of overwriting the file.

Turing – Level 1, High Level 1, Level 3

Turing is a programming language that supports both procedural and object-oriented programming styles. The following notes pertain to the Turing programming language:

- the *import* command loads a class for use in a program (e.g., in the Level 1 sample the student has imported the *GUI* class library to generate buttons);
- the dot operator, a period (*.*), can be used to access class behaviours (e.g., in the Level 1 sample, the statement *Draw.Cls* calls the clear screen behaviour of the *Draw* class,

and the statement *Time.Delay(3000)* calls the delay (3 seconds) behaviour of the *Time* class);

- all procedures must be declared before they can be used. Each procedure is blocked off by a descriptive method header and closing statement. For example, *proc ques13a ... end ques13a* defines the procedure *ques13a*.
- buttons are stored as Integers, and the parameters are *GUI.CreateButtonFull (x, y, width, text, procedure to be called, height, keyboard shortcut, default truth value)*;
- *locatexy(x,y)* sets the position of text displayed by the *put* command;
- *color (x)* sets the text colour displayed by the *put* command (e.g., in the High Level 1 sample, the student alternates the colours from black to green to red);
- *open*, *close*, and *assert* are used to control file access;
- *seek: filename, ** will move the file pointer to the end of the file.

C++ – High Level 2

C++ is a programming language that supports both procedural and object-oriented programming styles. The following notes pertain to the C++ programming language:

- the *#include* statement adds components of the language to the particular program (e.g., *#include <string>* adds the string class to the program);
- the statement *using namespace std*; groups a set of global classes, objects, and/or functions under a name. It is another way of adding features to the program and is used in conjunction with *#include* statements;

- the line *int main()* starts the main function; in the case of the High Level 2 sample, it is the only user-defined function in the program. The *main()* function is always the first to be executed in C++. In some programs, *void main()* is used instead of *int main()*. Use of *int main()* requires use of a *return* command;
- variables are declared by first stating the type and then one or more variable names, separated by commas (e.g., *string nzAnswer*; declares a string called *nzAnswer*);
- the statements *ofstream outFile* and *outfile.open("ErgonomicsRiskSurveyResults.txt")*; create a file object and link the file object to a name within the program. *outfile<<* is used to write data to the file. *outfile.close()*; closes the file;
- *cout <<* writes information to the screen. Variables and text are joined using the *<<* operator;
- *cin>>nEmployerNumber*; gets user data from the keyboard;
- control structures use braces to show their limits:

```
if (nResponse==1)
{
    statement(s)
}
```

Pascal/Delphi – Low Level 3, Level 4

Pascal is a programming language that supports both procedural and object-oriented programming styles. Delphi is an object-oriented language based on Pascal. There are many versions of Pascal and Delphi. The following notes are general guidelines:

- programs begin with the statement *program programName*; ;
- the statement *uses wincrt*; emulates a terminal-like coloured text screen in a Win32 GUI window;

- program statements are not case-sensitive;
- variables are declared in a *var* structure where the variable is listed followed by a colon and the variable type:

var

AnswerOne: integer;

- the statement *WriteLn(Please enter your workplace number:);* writes output to the screen;
- the statement *ReadLn(Worknum);* reads data from the keyboard;
- the statement *AssignFile (SurveyFile, 'G:\ERGORSLT.txt');* opens the file and assigns a program name to it;
- the statement *Append(Surveyfile);* sets the file mode to append;
- the statement *Write (Surveyfile, '0');* writes data to the file;
- the statement *Close(Surveyfile);* closes the file;
- code blocks are defined by *begin ... end;*
- structures such as *if* structures start with the keyword and condition and end with a semicolon:

if AnswerTwo = 1 then

begin

Write (Surveyfile, '0');

Total := Total + 0;

end ;

In this example the semicolon after the keyword *end* is the end of the *if* structure;

- the structure *Repeat ... Until ()* does not require a *begin ... end* structure to include multiple lines of code.

Java – Low Level 2, Level 2, High Level 3, Low Level 4, High Level 4

Java is an object-oriented programming language. The following notes pertain to the Java programming language:

- *import java.io.*;* loads the Java input/output class library for use in the program;
- *final* is the keyword used to declare a constant. Constants are often named in upper case (e.g., see High Level 3 sample, *final int MEDIUM = 5;*);
- Java has many different classes that can be used to read from the screen or from a file (e.g., *DataInputStream* or *BufferedReader*). Information regarding these classes can be found in the Java API at www.java.sun.com. In the Low Level 4 sample, a *KeyboardReader* class has been used, and in the Level 2 sample, an *In* class has been used. These classes have been provided to the students and developed outside of the Java API. There are many locally developed classes that could be used for user input;
- Java has many different classes that can be used to write information to a file (e.g., *PrintWriter* or *DataOutput*);
- opening and closing braces { } are used to separate the program into logical blocks;
- *for (int i=0; i<5; i++)* is the FOR-loop structure in Java, *for (initialization; condition; increment);*
- *System.out.println()* is a Java command to send data to a terminal window;
- an escape character can be used to perform formatting in the output. *\n* moves the output to the next line, and *\t* inserts a tab;

- `||` is the Boolean operation OR, and `&&` is the Boolean operation AND;
- `.charAt(0)` returns the character that is at the first position in the String (e.g., in the High Level 3 sample, the student checks if the user would like to continue `char check = in.readLine().charAt(0);`);
- `trim()` is a String method used to remove all leading and trailing white space from a String object;
- `Integer.parseInt()` returns the Integer value of the String argument (e.g., in the High Level 3 sample, `workstationNum = Integer.parseInt(in.readLine());`);
- the `try {...} catch (Exception e) {...}` structure is used to identify code areas where an exception might occur, handle that exception, and generate a message to the user that an exception has occurred.