University of Wollongong

School of Computer Science & Software Engineering

CSCI124 **Applied Programming** **Spring 2012**

Assignment 2 **(Due: Week 6, Wednesday 29 August)** **6 marks**

Aim:

Developing programs via functional decomposition and pointers.

On completion you should know how to:

- design a program by decomposing the problem into small functional units,
- implement a program using pointers and dynamic memory,
- produce a database program for maintaining records.

Requirements:

You are provided with an incomplete database program for accessing lost and found pets. Your task is to firstly convert memory management from static to dynamic memory and implement the main search function. Each record in the database contains the following information:

```
Status          // lost or found
Type            // dog or cat
Gender          // male, female or unknown
Breed           // no more than 20 chars
Age             // in years and mths. (-1 if unknown)
Colour          // char[] type
Location        // suburb where found or lost
Phone No.       // Ph. of person to contact
```

The program is implemented in 3 files: **main.cpp** contains the text-menu based user interface. **ass2.h** contains public function prototypes. **ass2.cpp** contains the function definitions of the database program. Test data is provided in **pets.txt**.

Step 1: involves converting the program from static to dynamic memory. To do this, first convert the DB array into an array of pointers to PetRecords:

```
PetRecord *gPetRecs[cMaxRecs];
```

Then modify the ReadFile() function so that as records are read, dynamic memory is allocated to array elements using the new operator:

```
gPetRecs[i] = new PetRecord;
```

This statements should go just after the break statement within in the read file loop. Now convert all references to the PetRecord fields to '->'. e.g.

```
gPetRecs[i]->Status = Lost;
```

Step 2: involves implementing an AddRecord() function for adding a new record to the DB:

a adds a new record to the file
Example:

```
Command >   a
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+               Add Record to Database                +
+              (Enter your pets details)              +
+         (Enter ? or -1 if detail is unknown)        +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
Is the pet lost or found?   (l/f) => l
Is the pet a dog or cat?    (d/c) => d
What breed is the pet?            => Border Collie
What age is the pet?     (yy mm ) => 5 6
What colour is the pet?           => black and white
In what suburb was the pet lost?  => Corrimal
What is your phone number?        => 42123412
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+     A new record has been added to the database     +
+          45 records are in the database             +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Note: all character data should be stored as lower case. The new record should be both added to the gPetRes[] array and the data file. To append the record data to the end of the data file use: fout.open(filename, ios::app); ***// opens file in append mode.***
Unknown data fields are represented with the word "unknown" or a -1 integer.

Step 3: involves implementing a Search() function for searching the database. The search function should request the pet details from the user and remove (delete) any non-matching records from the gPetsRecs[] array. Thus to perform a search the user first selects the "(r)ead data file" menu option from the main menu, then performs a search and displays the remaining records. If too many records match the search, the user can search the remaining records with more details entered to reduce the number of matching records.

s searches the array and removes records that do not match the search criteria
Example:

```
Command >   s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                  Search Database                    +
+             (Enter your pets details)               +
+         (Enter ? or -1 if detail is unknown)        +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
Search lost or found pets?  (l/f) => l
Search for a dog or cat?    (d/c) => d
What gender is the pet?           => ?
What breed to search for?         => Border Collie
What age to search for?  (yy mm ) => -1 -1
What colour to search for?        => black and white
What suburb was the pet found?    => ?
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
+          There are  3 pets in the database          +
+           that match the search criteria            +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Further information may be provided on the messages web page. Do not alter main.cpp or ass2.h as these are not submitted. Make sure your code continues to work after each step before proceeding onto the next step. If you are unsure of exactly what to do in each step do not hesitate to ask your tutor or demonstrator. (The onus is on you to resolve anything you find unclear or ambiguous).

Submit:

Before submitting your ass2.cpp file check the format of your source files to ensure tabs and newlines appear correct on UNIX. Submit your **ass2.cpp** file using the submit facility on UNIX ie:

```
$ submit –u login –c CSCI124 –a 2 ass2.cpp
```

where 'login' is your UNIX login ID.

Deductions will be made for untidy work or for failing to comply with the submission instructions. Requests for alternative submission arrangements will only be considered before the due date. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline must be made to the Subject Coordinator before the submission deadline. Supporting documentation must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the points awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than four days late.