McGill

April 2012
Final Examination

Introduction to Software Systems

COMP-206

April 20, 2012 at 9:00 – 12:00

Examiner: Joseph Vybihal Assoc Examiner: Mike Langer

Student Name:		**McGill ID:**									

INSTRUCTIONS:

- This is a **CLOSED BOOK** examination.
- You are permitted **TRANSLATION** dictionaries ONLY.
- **STANDARD CALCULATOR** permitted ONLY.
- This examination is **PRINTED ON BOTH SIDES** of the paper
- This examination paper **MUST BE RETURNED**
- You are permitted to write your answers in either **English or French**
- Write your answers in the **exam booklet** providedd
- Attemp all questions, **part marks** will be assigned, show your work.

Grading

Section	Grade	Your Mark
Question 1: Definitions	10	
Question 2: Linked Lists	20	
Question 3: HTML and CGI	10	
Question 4: C & CGI	20	
Question 5: Python	10	
Question 6: Make and Modules	10	
Question 7: GDB	20	
Total	100 %	

Question 1: Definitions

Answer these questions in the line provided. Do not exceed the line!

1. The GNU Profiler is

   _____

2. Typing “finger” by itself on the command-line and then pressing enter

   _____

3. What is the “extern” statement in C used for:

   _____

4. What is RCS:

   _____

5. What is a Session?

   _____

Question 2: Write a Function to Reverse a Linked List

Assume that you are given the reference to the beginning of a linked list and that the linked list is already created in memory. This list can either be empty or contain a list of nodes. The linked list is a singly linked list with the following definition. Create a function named "reverse" that when called reverses the linked list passed to it.

```
struct linked_list
{
     int data;
     linked_list *next;
};
```

Note:

1) You may not modify the original linked list passed as a parameter to your function. Make sure you return a newly created linked list, which is the reverse of the original.
2) Add error-handling code in your procedure.

Question 3: HTML and CGI

Create a home page to a Google knock-off search engine. Write only the HTML and CGI code for an index.html file that displays a graphic named "doogle.jpg" centred at the top of the page. Below that, also centred, in large blue font display "Search to your heart's content!". Under that is a long text box that permits the user to input a maximum of 80 characters. The box should be wide enough to display 50 of those 80 characters. Underneath the text box are two buttons and one link. The first button is titled "Search", the second button is titled "Feeling Lucks", and the last link displays "Français" (with the correct ASCII for ç) linking to a French version of this page called "index-f.html" (you are not writing the French version of the page). The search button invokes a C program called "search.cgi". The feeling lucks button invokes a C program called "lucky.cgi".

Question 4: CGI and C Programming

This question is connected to Question 3, above. Implement the "search.cgi" program. Assume this web site has a CSV file called web.csv. Each row of this CSV file has two fields. The first field is the name of a web site, like "**www.site.com**". The second field is a sentence describing that web site, like "An animal store". Your C program will display on the user's browser all the web site names matching the search criteria they input in Question 3 as it compares to the sentence description in the second field. The displayed web site names must be click-able links. You can use the string.h library to help you. You must tokenize the input search criteria and count the number of words that match. If it is greater than 60% then display the link. For example, search criteria: "Dog or animal store". Tokens: "dog", "or", "animal", and "store". A description field from the CSV: "An animal store". Number of matching words: 2, "animal" and "store". Ratio: 2/3. This is greater than 60% so display the link.

Question 5: Python Programming

Answer the following two questions:

(1) What does this Python program display?
(2) Describe, in short, what each line of this program does

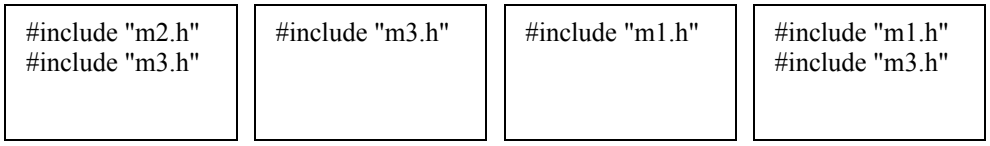
```
import sys
if __name__ == "__main__":
        if len(sys.argv) > 1:
                try:
                      f = open(sys.argv[1], "rb")
                except:
                      print "No file named %s exists!" % (sys.argv[1],)
                while 1:
                       t = f.readline()
                       if t == '':
                           break
                       if '\n' in t:
                           t = t[:-1]
                       if '\r' in t:
                           t = t[:-1]
                       sys.stdout.write(t + '\n')
                f.close()
```

Question 6: The Make Program and modular programming

Assume that a student has the following programming practises:

i. There are always one .h file and one .c file (of the same name) in a module. For example, a module called "mod" would have "mod.h" and "mod.c";
ii. The corresponding .h file is always included in the .c file. No other includes, except the standard C includes (e.g. <stdio.h>, etc.), is allowed in .c files.
iii. The .h files may include other .h files.

This student is currently writing, for her assignment, a program of 4 modules: m1, m2, m3 and main. She would like to name her executable "ass3". Here are the "include" directives found in the four .h files:



This student does not use any tool to manage program compilation.
The output of "ls –l" is as follows:

```
total 13
-rw------- 1 student   734 Nov 17 10:02 m1.c
-rw------- 1 student   203 Nov  2 11:39 m1.h
-rw------- 1 student  2823 Nov 17 10:03 m1.o
-rw------- 1 student   425 Nov 16 20:21 m2.c
-rw------- 1 student   220 Nov  2 11:39 m2.h
-rw------- 1 student  2213 Nov 13 12:24 m2.o
-rw------- 1 student   323 Nov 16 22:38 m3.c
-rw------- 1 student   212 Nov  8 11:39 m3.h
-rw------- 1 student  2533 Nov 16 22:40 m3.o
-rw------- 1 student  3213 Nov 17 12:27 main.c
-rw------- 1 student   243 Nov  8 11:39 main.h
-rw------- 1 student 24745 Nov 10 11:39 main.o
-rwx------ 1 student 29234 Nov 10 11:39 ass3*
```

From the above information, answer the following questions:

1. The student decides to use the "make" tool to facilitate project management. Write a makefile for her that gives an executable of her program when "make" is issued, and clears all intermediate files when "make clean" is issued. The makefile must always work in the **optimal** way (i.e. compiles only the files that require compilation)
2. Now the student does a "make" with her makefile. What will be the output of "ls –l" after this make? Giving only the file names and times would suffice. (Current time is assumed to be Nov 17 14:00 and the compile take 2 seconds)

Question 7: Debugging and Logic Error

Answer the following questions about GDB:

(a) Write a command to run gdb without printing the front material, which describes GDB's non-warranty.
(b) What does gdb -h do?
(c) Look at the following code and answer the following questions:

```
1. int main() {
2.    int i;
3.    printf("welcome");
4.    for (i=0;i<10;i++)
5.    i - = 1;
6.    printf("end of program");
7.    return i;
8. }
```

1. How do I set a break point to stop before entering the main program?
2. What happens if I write step three times, i.e. one after the other after stopping in main?
3. What happens if I write next three times after stopping in main?
4. Give a command to set a break point at line 7.
5. Give a command to start execution from the previous break point.
6. What happens after executing the command in question 5?
7. There is a logic error in this program. Assume that the programmer does not realize this and is now using debugger to determine what the error is. In diagnosing this logic error what would the programmer do (and see) when using gdb? How can the logic error be corrected?