Dalhousie University
CSCI 2132 — Software Development
Winter 2018
Final Examination
April 14
12:00pm-3:00pm

Student Name:	____________________
Student ID Number:	____________________
FCS Username (CSID):	____________________
Signature:	____________________

Instructions (Read Carefully):

1. Aids allowed: one 8.5" by 11" piece of paper with anything written or printed on it (both sides). No textbooks, computers, calculators, or other aids.
2. This exam booklet has 15 pages, including this page. Ensure that you have a complete paper.
3. Understanding the exam questions is part of the exam. Therefore, questions will **not** be interpreted. Proctors will confirm or deny errors or ambiguities only.
4. Marks will not be given for documentation, though feel free to add documentation if you feel necessary. You however should make your code readable to facilitate marking.
5. The blank sheet given with this exam book is a piece of scratch paper. Do not submit.

Question	1	2	3	4	5	6	7	8	9	10	Total
Marks	12	12	9	9	12	7	8	10	8	13	100
Scores											
Marker											

1. [12 marks] True-false: 2 marks each. No justification necessary.

   (a) A linker for the C programming language modifies the source program by obeying directives.
   False

   (b) When we open a binary file for reading use a C program, the mode string used in the `fopen` function call should be `"r"`.
   False

   (c) The command `ls` is an external command of the shell.
   True

   (d) In the C programming language, the following two pairs of `scanf` format strings are equivalent:
   `"%c"`
   `" %c"`
   False

   (e) The function `malloc` allocates memory from the heap (free store).
   True

   (f) The function `fprintf` can be used to print data to the standard output.
   True

2. [12 marks] Multiple-choice – no justification necessary. Circle the *single* best answer. 2 marks each.

   (a) Suppose that a student wrote a Bash shell script named `foo.sh`. At present, its permission, when expressed as an octal number, is `644`. When this student ran the command `./foo.sh`, he got the following error message:

   ```
   -bash: ./foo.sh: Permission denied
   ```

   This student would like to use a command to change the permissions, so that he will be able to run this shell script as the file owner without getting the above error message. Which one of the following four commands will NOT complete this task successfully? Note that the three commands that would work do not grant the same permissions.

   A. `chmod uo+x foo.sh`
   [✓ B]. `chmod 444 foo.sh`
   C. `chmod ug+x foo.sh`
   D. `chmod 500 foo.sh`

   (b) Assume that the pathname of a user's home directory on the bluenose server is `/users/cs/grant`. He created a directory named `csci2132` in his home directory, and then created three subdirectories named `a1`, `a2` and `a3` in the directory `csci2132` that he created. He then changed his current working directory to the `a2` directory that he created. Which of the following four pathnames does NOT refer to the `a3` directory that he created?

   [✓ A]. `../csci2132/a3`
   B. `./../../csci2132/a3`
   C. `/users/cs/grant/csci2132/a3`
   D. `$HOME/csci2132/a3`

   (c) Suppose `a` is a one-dimensional `int` array and `p` is an `int` pointer. Assume that the assignment `p = a` has just been performed. Which one of the following expressions is illegal because of mismatched types?

   A. `p == &a[0]`
   B. `*p == a[0]`
   [✓ C]. `p == a[0]`
   D. `p[0] == a[0]`

(d) Which of the following statements regarding `gdb` is INCORRECT?

A. `gdb` is a symbolic debugger.

B. The `gdb` command `break` can be used to set a breakpoint.

C. The `gdb` command `print` can be used to print the value of a variable when the program pauses at a breakpoint.

[✓ D]. The two `gdb` commands `next` and `step` are equivalent.

(e) Suppose that we would like to write a C program to read from a text file named `hello.txt` and print its content to the screen. To do this, we use a file pointer `fp`. Which of the following four statements should we use to open this file?

[✓ A]. `fp = fopen("hello.txt", "r");`

B. `fp = fopen("hello.txt", "w");`

C. `fp = fopen("hello.txt", "a");`

D. `fopen(fp, "hello.txt");`

(f) Suppose that the following declarations are in effect:

```
int a[] = {17, 14, 10, -5, 14, 19};
int *p = &a[1];
int *q = &a[4];
```

Which of the following four logical expressions evaluates to false?

A. `p < q`

B. `*p == *q`

C. `a < p`

[✓ D]. `p == q`

3. [9 marks] Show the output produced by each of the following programs. You need not justify your answers.

   3 marks each.

   (a)
```
#include <stdio.h>

void swap(int, int);

int main(void) {
  int a = 4;
  int b = 5;

  swap(a, b);

  printf("%d %d\n", a, b);
}

void swap(int a, int b) {
  int temp = a;
  a = b;
  b = temp;
}
```
```
4 5
```

(b)
```
#include <stdio.h>

int main(void) {
  int i = 4, j = 8;
  int *p = &i;
  int *q;
  int *r = &j;

  *r = *p;
  q = p;
  (*q)--;

  printf("%d %d %d\n", *p, *q, *r);

  return 0;
}
```
```
3 3 4
```

(c)
```
#include <stdio.h>

int main(void) {
  char s[] = "software development";

  printf("%c\n", s[1]);
  printf("%s\n", s+9);

  *(s+8) = '\0';
  printf("%s\n", s);

  return 0;
}
```
```
o
development
software
```

4. [9 marks] For each of the following questions, develop a single command line to perform the task required. You need not justify your answer.

   To receive any mark for each question, you are only allowed to use commands from the following list: `ls`, `wc` and `grep`.

   3 marks each.

   (a) Print a list of files in the directory `/usr/bin` whose names start with the English word "git". The file named `git` is considered one of these files.

   ```
   ls /usr/bin/git*
   ```

   or

   ```
   ls /usr/bin | grep ^git
   ```

   (b) Print out the number of files in the directory `/usr/bin` whose names start with character `u`, `n`, `i` or `x`, do NOT end with and `u`, `n`, `i` or `x`, and are 4 characters in length.

   For example, `/usr/bin/uniq` is such a file, but `/usr/bin/users` and `/usr/bin/neqn` are not.

   ```
   ls /usr/bin/[unix]..[!unix] | wc -l
   ```

   or

   ```
   ls /usr/bin | grep ^[unix]..[^unix]$ | wc -l
   ```

   (c) Print out the number of words in the Linux dictionary file `/usr/share/dict/linux.words` containing the word `soft` (that is, the word `soft` appears in it as a substring) and do not contain any hyphens.

   For example, `soft` and `unsoftened` are such words, but `soft-voiced` is not.

   Recall that this dictionary file contains one English word per line.

   ```
   grep ^[^-]*soft[^-]*$ /usr/share/dict/linux.words  | wc -l
   ```

5. [12 marks] Give concise answers to the following questions. 4 marks each.

   (a) The waterfall model of software development life cycle has five steps. Write down the names of four of these steps. You need not explain what these steps do.

   Solution: You can write down four of the five below (some steps have multiple names and you just have to write down one name): requirement analysis, software design (or design), implementation (or coding), verification (or testing) and maintenance (if you write down patching it is also fine).

   (b) We learned that the `make` utility supports separate compilation of a program consisting of multiple modules. Describe one advantage of separate compilation.

   Solution: If we modify one source file, then only those targets that depend on this source file have to be re-compiled.

   (c) The following code snippet is supposed to create an identical copy of a string and returns a pointer that points to it, but it is incorrect. Find out what is wrong, explain why it is incorrect, and fix the error by modifying the code printed below.

```
char *duplicate(char *p) {
  char *q;

  strcpy(q, p);
  return q;
}
```

   Solution: `q` stores an arbitrary memory address. That is, no memory has been allocated to store the identical copy. To fix this, add the following code before the `strcpy` function call:

```
q = malloc(strlen(s) + 1);

if (q == NULL)
  return NULL;
```

6. [7 marks] This question asks you to describe how you would write a multiple-file C program to solve a problem.

   Do not write actual C code for this problem. Instead, you are asked to describe how to divide the program into multiple files, and then write a makefile for separate compilation.

   The program reads multiple lines from `stdin` and stops when reading an empty line. Each time it reads a line, it inserts the content of this line (which is a string) into the head of a linked list. After that, the program prints all the lines using the linked list, so that the lines of the input are printed in reverse order, i.e., the last line is printed first.

   (a) Show how to use two C source files plus one C header file to organize your program. To describe your solution, for each source or header file, give it a descriptive name and use one sentence to describe the main purpose of this file. You need not give any other details.

   reverselines.c: the main function (main.c is also fine)

   linkedlist.c: the functions for operations over linked lists

   linkedlist.h: to share the prototypes of the functions for linked lists

   (b) Write a makefile to compile your program. Your makefile must support separate compilation. Indent your solution to indicate the tab characters.

```
reverselines: reverselines.o linkedlist.o
      gcc -o reverselines reverselines.o linkedlist.o

reverselines.o: reverselines.c linkedlist.h
      gcc -c reverselines.c

linkedlist.o: linkedlist.c linkedlist.h
      gcc -c linkedlist.c
```

7. [8 marks] Write the following function (NOT a complete program):

```
void find_two_smallest(int a[], int n, int *smallest,
                       int *second_smallest);
```

When passed an array `a` of length `n`, this function will search for its smallest and second smallest elements and store them in the variables pointed to by `smallest` and `second_smallest`, respectively.

To receive full marks, you are required to use a pointer to access array elements. Up to two marks will be deducted if you use array subscripting in your function body (still, better than no solution).

To simplify your solution, you can assume that the array `a` has at least two elements and no error handling is required.

```
void find_two_smallest(int a[], int n, int *smallest, int *second_smallest) {
  int* p;

  if (*a < *(a+1)) {
    *smallest = *a;
    *second_smallest = *(a+1);
  } else {
    *smallest = *(a+1);
    *second_smallest = *a;
  }

  for (p = a + 2; p < a + n; p++)
    if (*p < *smallest) {
      *second_smallest = *smallest;
      *smallest = *p;
    } else if (*p < *second_smallest)
      *second_smallest = *p;
}
```

8. [10 marks] Two words are anagrams if they are permutations of the same letters. For example, `smartest` and `mattress` are anagrams, but `apple` and `lemon` are not.

   Write a **complete** C program that tests whether two words are anagrams. The following is an example of how the program works:

   ```
   Enter first word: smartest
   Enter second word: mattress
   The words are anagrams.
   ```

   To simplify your program, you can assume that these two words contain lowercase letters only. That is, the user always input one word containing lowercase letters when asked for one word.

   Use the white space below as well as the next page to write down your solution.

   Hint: The following is a possible implementation (you are not required to follow it, but this is the easiest way of doing this): Write a loop that read the first word, character by character, using an array of 26 integers to keep track of how many times each letter has been seen.

   Use another loop to read the second word, except this time decrementing the corresponding array element as each letter is read.

   After the second word has been read, use a third loop to check whether all the elements in the array are zero. If so, the words are anagrams.

   The above approach does not even use C strings!

   ```
   #include <stdio.h>

   int main(void) {
     char ch;
     int count[26] = {0};
     int i;
     int result = 1;

     printf("Enter first word:");

     while ( (ch = getchar()) != '\n')
       count[ch-'a']++;

     printf("Enter second word:");

     while ( (ch = getchar()) != '\n')
       count[ch-'a']--;
   ```

```
  for (i = 0; i < 26; i++)
    if (count[i] != 0) {
      printf("The words are not anagrams.\n");
      return 0;
    }

  printf("The words are anagrams.\n");
  return 0;
}
```

9. [8 marks] Write a utility called `duplicate.sh` satisfying the following specification:

   `Utility: ./duplicate.sh [-l] {directory_name}`

   This utility copies all the regular files in directory `directory_name` into a directory called `.duplicate` in your home directory.

   If `.duplicate` does not exist, the utility creates this directory before copying files.

   When `-l` option (this is the letter `l`) is present, the current content of `.duplicate` is listed. When this option is present, no `directory_name` is specified as command-line argument.

   Complete this script by filling in the blank lines in the program given below. Do not put multiple statements on the same line.

```
#!/bin/bash

# check what command-line argument the user inputs

case $1__________ in
     # list the content of the .duplicate folder in home directory
     "-l")
          ls ~/.duplicate
          ;;

     # copy all the files in the directory specified as command-line
     # argument by the user to the .duplicate folder in home directory
     *)
          # handle the case in which the directory
          # specified by user does not exist

          if [ _!__-d_________ ~/.duplicate ]; then
             mkdir ~/.duplicate
          fi

          for file in `ls $1`___________
          do

             cp $1/$file ~/.duplicate/$file___________
          done
esac
```

It's okay to use -e instead of -d. There is also an alternative solution to the last two blanks.

10. [13 marks] Assume that we have the following structure that defines nodes in a linked list:

```
struct node {
  int value;
  struct node *next;
};
```

(a) Implement the following function (NOT a complete program):

```
int count_less(struct node *list, int val);
```

The `list` parameter points to the head of a linked list. The function should return the number of nodes in the linked list whose `value` member is less than parameter `val`.

```
int count_less(struct node *list, int val) {
  int count = 0;

  while (list != NULL) {
    if (list->value < val)
      count++;

    list = list->next;
  }

  return count;
}
```

(b) Implement the following function (NOT a complete program):

```
struct node* insert_mtf(struct node* list, int val);
```

The `list` parameter points to the head of a linked list. This function first searches this list to find out whether there exists one node whose `value` member is equal to `val`. If such a node is found, then this node is first removed from the list and then inserted to the head of the list (this is called move-to-front). Otherwise, allocate a new node to store the value of `val`, and insert it to the head of the list. The function then returns a pointer to the head of the linked list.

For example, suppose that we use this function to insert the following values into an empty linked list: $8, 9, 1, 2, 5, 8, 5, 0$. Then, the `value` members of the nodes in the linked list starting from the head store $0, 5, 8, 2, 1, 9$.

You can assume that there are no two nodes in the list whose `value` members are equal.

```
struct node* insert_mtf(struct node* list, int val) {
  int found = 0;
  struct node *prev, *curr;
  prev = NULL;
  curr = list;

  while (curr != NULL) {
    if (curr->value == val) {
      found = 1;
      break;
    }

    prev = curr;
    curr = curr->next;
  }

  if (found) {
    if (prev == NULL)
      return list;

    prev->next = curr->next;
    curr->next = list;
    return curr;
  }
  else {
    curr = malloc(sizeof(struct node));

    if (curr == NULL) {
      printf("out of memory\n");
      exit(EXIT_FAILURE);
    }

    curr->next = list;
    curr->value = val;
    return curr;
  }
}
```