CMPUT 201 — Winter 2000
Practical Programming Methodology
Section X10 (Beck)
Final Exam (Exam A)
Tuesday April 11, 2000

Name:

Id:

Instructions

- Write your name and student id number in the boxes on the first page.
- This is an **OPEN BOOK** exam. Programmable electronic devices are **NOT** allowed to be used during this exam. Time allowed: 120 minutes.
- Place all answers in the spaces provided on the question pages. **JUSTIFY** each answer appropriately.
- This exam counts 32% toward your final grade in this course. This exam is marked out of 57 marks. The weight of each question is indicated in square brackets by the question number.
- This exam is not impossible, but the questions do not necessarily have obvious answers. Read each question thouroughly and think about each question for couple of minutes before answering it.
- Some questions require you to write C or C++ code. You will not be penalized for minor syntactical errors or banal omissions, but I must be able to understand your code.
- When you are given code in this exam, you may assume that the code you are given will compile.
- There should be 15 questions and 17 pages in this exam booklet. You are responsible for checking that your exam booklet is complete.

Question	Mark	Out Of
1		3
2		3
3		2
4		3
5		5
6		3
7		5
8		5
9		4
10		2
11		3
12		7
13		4
14		3
15		5
TOTAL		57

Question 1 [3 marks]: What are the possible outputs of the following C program. Provide a short explanation of what you believe happens to produce each possible output.

```
#include <strings.h>
#include <stdio.h>

char * reverse (char *fred) {
   int i, length;
   char tmp;
   length=strlen(fred);
   for (i=0; i < (length)/2; i++) {
       tmp=fred[i];
       fred[i]=fred[length - i];
       fred[length - i] = tmp;
   }
   return(fred);
}

int main() {
   char geekazoid[80] = "TinkyWinky";
   printf("%s\n", reverse(geekazoid));
   exit(0);
}
```

Question 2 [3 marks]: I have written a Class named Bob in C++. Class Bob is implemented in the file `bob.cc`. The interface to Class Bob is in the file `bob.h`. As part of it's private data, class Bob uses another class named Beck, implemented in the file `beck.cc` with the interface in file `beck.h`. I have written a main program in the file `fatboy.cc`. This main program uses class Bob. Write a Makefile where the default target will correctly compile a program named `fatboy` from the above sources. Be sure all dependencies are correctly allowed for and no extra recompilation is done if any files are changed.

Question 3 [2 marks]: What's wrong with the C library functions `strcpy` What function should we use instead to avoid the problem?

Question 4 [3 marks]: What is the output of the following C program?

```
#include <stdio.h>

void recur(int n) {
  if ( n ) {
    printf ("%d ", n);
    recur ( n / 2 );
  }
}

int main() {
        recur(30);
putchar('\n');
        exit(0);
}
```

Question 5 [5 marks]: Explain what happens when the following program is run using the input on stdin of a file containing one line of input: `hello`

```
#include<stdio.h>

char * mysub(char * s) {
  int unused = 0;
  static char tmp_buf[20];
  int len, i;

  memset(tmp_buf, '%', 20);
  len = strlen(s);

  for (i=0; i<len; i++) {
    tmp_buf[len - i] = s[i];
  }
  return(tmp_buf);
}


int main() {
  int len;
  char my_buf[20];

  while (fgets(my_buf, 20, stdin) != NULL) {
    len = strlen(my_buf);
    my_buf[len-1]= (my_buf[len-1] == '\n') ? '\0' : my_buf[len - 1];
    printf("%s", mysub(my_buf));
    putchar('\n');
  }
}
```

Question 6 [3 marks]: Write the following function in C or C++;

```
int * resize(int *arr, int oldsize,  int *newsize)
```

function `resize` takes array `arr` and resizes it to hold the number of elements in the int pointed to by `newsize` (making it bigger or smaller and preserving the contents of the old array). It returns a pointer to the start of the new (possibly relocated) array, and stores the new size of the array in the integer pointer to by `newsize`. If it is not possible to resize the array to the new size the original array should be left unmodified (and the old size would then be stored in `newsize`.)

Question 7 [5 marks]: Write a program in C that will read in an entire file of input from standard input, and produce output that is the entire input file, followed by the entire input file again. Be sure your program places no artificial limits on the length of the input beyond the amount of memory and disk available to you.

Question 8 [5 marks]: Consider the following structure, and array:

```
struct symtab {
   char *symbol;
   short address;
};

struct symtab labels[100];
```

Write a C subroutine that sorts the array `labels` (assume it is a global variable) into ascending order based on the value of the `address` field, then prints out the array in sorted order printing out the symbol string followed by the address as a regular decimal number, one per line.

Question 9 [4 marks]: On running your C program, your program stops with the message `Segmentation Fault (core dumped)` .

Part 9 a [2 marks]: What causes a Segmentation Fault? Give an example of something we could do to cause one.

Part 9 b [2 marks]: Show how you would find where the program dumped core using the `gdb` debugger.

Question 10 [2 marks]: Give a UNIX command that would help us find out what header files we needed to include to use the C library function `getopt`.

Question 11 [3 marks]: Provide all the unix shell commands would you use to update your checked out copy of assignment 3 from the code in the repository, so that CVS would merge the changes made into the repository back into your files.

Question 12 [+7 to -7 marks]: RIGHT - WRONG True False. For each statement below, write "True" if you know it to be true, "False" if you know it to be false, or do not write anything if you do not know the answer. You will be receive 1 mark for each correct answer, -1 mark for each incorrect answer. You will receive 0 marks for any question which is not answered, So, if you are not certain of your answer, leave the question blank and you will not lose any marks. This makes it possible to get -7 marks on this question, which will subtract from your final total on this exam.

Part 12 a []: To see the past commit messages for a file `fred.c` checked out of CVS in the current directory, I could give the command: `cvs log fred.c`

Part 12 b []: If I have changed some files which were checked out of CVS in my current directory, the command `cvs update` will commit all changed files in the current directory back to the repository.

Part 12 c []: With a dynamically allocated null-byte terminated C string, the function `sizeof()` returns the number of characters in the string, not including the 0 byte on the end.

Part 12 d []: With a statically allocated array of char, holding a null-byte terminated C string, the function `sizeof()` returns the number of bytes of storage currently allocated.

Part 12 e []: In C++, if Class B declares class A as a `friend` then class A can see the private data elements of class B.

Part 12 f []: In C and C++, the statements `A[i]` and `*A+i` are exactly equivalent.

Part 12 g []: In C++, If we allow an object to be bitwise copied, it won't be harmful if the object does not contain any pointer data members.

Question 13 [4 marks]: Assume you have an array declared as follows:

```
unsigned char Memory[32767]
```

Show the code with proper casting to store a `short` into the array starting at index 10 and to retrieve a `short` from the array at index 20.

Question 14 [3 marks]: If I allocate an array in C++ as follows:

```
int * fred;
fred = new int [100];
```

Will anything bad happen if I free this integer array with the statement `delete fred`? Explain your answer.

Question 15 [5 marks]: What is the output of the following program:

```
#include <iostream.h>

class Geek {
public:
  Geek (int i = 99) { geek = i; cout << "default, id="<< geek;
                      cout << "\n"; };
  Geek (Geek const & other ) { geek = other.geek;
                               cout<<"copy from id=" << geek;
                               cout << "\n"; };
  ~Geek () { cout << "destroying id=" << geek << "\n"; };

  Geek const & operator =(Geek const & other);

  int getvalue(void) {return(geek);};

private:
  int geek;
};

Geek const & Geek::operator =(Geek const & other) {
  if (this != &other) {
    cout << "assigning geek " <<  other.geek <<  " over top of geek " << geek
         << "\n";
    geek = other.geek;
  }
}

Geek myfunc(Geek ageek) {
cout << "Hello this is myfunc with a geek of value "
<<  ageek.getvalue() << "\n";
return(ageek.getvalue() - 5);
}

int main () {
  Geek Beck(42);
  Geek Christou;
  Geek Thompson=Beck;

  cout << "Starting Main\n";
  Beck = 31;
  Christou = myfunc(Beck);
  Geek Tullia = myfunc(Christou);
  cout << "Leaving Main\n";
  return(0);
}
```