CSE 303, Spring 2009
Final Exam
Wednesday, June 10, 2009

Personal Information:

Name: _____

Student ID #: _____

- You have 110 minutes to complete this exam.
  You may receive a deduction if you keep working after the instructor calls for papers.
- This exam is open-book/notes. You may not use any computing devices including calculators.
- Code will be graded on proper behavior/output and not on style, unless otherwise indicated.
- On C/C++ programming problems, you do not need to `#include` any libraries.
- Do not abbreviate code, such as "ditto" marks or dot-dot-dot ... marks.
- If you enter the room, you must turn in an exam before leaving the room.
- You must show your Student ID to a TA or instructor for your exam to be accepted.

Good luck!

Score summary: (for grader only)

Problem	Description	Earned	Max
1	C Pointers		15
2	C Programming		15
3	Compilation Mgmt. (Makefiles)		15
4	Version Control (Subversion)		10
5	C++ Parameter Passing		15
6	C++ Classes and Inheritance		15
7	C++ Programming		15
TOTAL	**Total Points**		**100**

1. C Pointers

What is the output of the following C program? (The lines that lead to the printing of output are bolded.) Assume that the program is running on a 32-bit computer.

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

void showarray(int* a, int l);

int main(void) {
    char s1[16] = "cse123";
    char* p1;

    int a[8] = {3, 6, 9, 12, 15};
    int* x = (int*) calloc(8, sizeof(int));
    int* p2;
    int i;

    showarray(a, 8);
    showarray(x, 8);

    for (i = 0; i < 8; i++) {
        x[i] = a[i] << 2;
    }
    showarray(x, 8);

    p2 = &a[2];
    p2 = p2 + sizeof(int);
    (*p2)++;
    p2--;
    (*p2) += sizeof(int);
    showarray(a, 8);

    p1 = s1;
    while (*p1) {
        (*p1)--;
        p1++;
    }
    printf("%s\n", s1);

    return 0;
}

// prints the given number of elements of the given array of integers
void showarray(int* a, int l) {
    int i;
    for (i = 0; i < l; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
```

2. C Programming

Suppose that you have a structure type with the following declaration:

```
typedef struct Node {
    char* str;
    struct Node* next;
} Node;
```

Write a C function named `concatenate` that accepts a pointer to a `Node`, the first node in a null-terminated list of nodes, and builds and returns a single string containing the concatenation of all the strings in the list.

For example, if you have a list that stores the following strings:

`"hello"` → `"how are you?"` → `" CSE 303 is great!"` → `"I agree"`

Then your method should return the string `"hellohow are you? CSE 303 is great!Iagree"`.

Your method should also **free all of the memory** associated with the list. That is, you should free all of the nodes themselves and also free the strings stored in each node.

If the node pointer passed to your method is null, return a null result. Assume that no string in any node is null.

You may assume that there is enough available memory on the system to accommodate any memory allocation you may need to do. Your function should not leak any memory; if you remove any blocks of memory other than the final block that contains your string to return, you must free them. You may use any string library functions you like.

---

3. Compilation Management (Makefiles)

Suppose you have the following collection of files that include each other in the following ways. Assume that ***file***.h is the header that corresponds to ***file***.c.

```
/* file1.h */
#include <stdio.h>

/* file1.c */
#include <stdlib.h>
#include "file1.h"

/* file2.h */
#include <stdio.h>
#include "file3.h"

/* file2.c */
#include <string.h>
#include "file2.h"
```

```
/* file3.h */
#include "file1.h"

/* file3.c */
#include <stdio.h>
#include <string.h>
#include "file3.h"

/* main.c */
#include <stdio.h>
#include <stdlib.h>
#include "file2.h"
#include "file3.h"
```

Below, write a complete `Makefile` with rules to build each appropriate source file into an object (`.o`) file (using `gcc` with whatever flags are appropriate) with proper dependencies so that a file will be recompiled if and only if a file it depends on is modified. Include a rule to build an executable `main` that runs the main function from `main.c`. This should be the default rule that runs if the user types `make`. Lastly, include a rule named `clean` that will remove all .o files and the `main` executable.

---

4. Version Control (Subversion)

Suppose that a pair of project partners, Alice and Bob, are working on the files from the previous problem #3. Alice would like to set up a Subversion repository to manage these files. She has the following files in her local directory:

file1.h	file1.c	file1.o	file2.h	file2.c	file2.o	file3.h	file3.c	file3.o	main.c	main.o	main	Makefile

a) Which of the files should be checked in to the SVN repo? **Circle the ones that should be checked in**, and/or put an X through any file that should *not* be checked in.

b) Suppose that the files you circled are checked in to the repo (revision 1), and then the following actions occur (in chronological order from top to bottom):

- Alice checks out all files from the repo.
- Alice modifies her `file1.c`.
- Alice modifies her `file2.h`.
- Alice commits `file2.h` to the repo.
- Alice modifies line 11 of her `Makefile`, the rule that builds `main`.
- Bob checks out all files from the repo.
- Bob creates new files `file4.c` and `file4.h` on his local copy.
- Bob adds a `Makefile` rule to build `file4.o` and modifies its line 11, the `main` rule, to include `file4.o`.
- Bob modifies his `file2.h` so that it includes `file4.h`.
- Bob commits `file2.h` to the repo.
- Alice runs an SVN `update`.
- Alice commits all uncommitted files she has modified (`file1.c` and `Makefile`).
- Alice and Bob both attempt to build their local copies by running `make`.

After the above steps, some problems have been introduced both in the repo and in the partners' local copies. A "problem" would be a file that is out of date, missing, broken, having a conflict, etc. **Describe the problems introduced, at which step(s) they occur or will be noticed, and how Alice and Bob can repair them.**

---

5. C++ Parameter Passing

Write the output of the following C++ code, which uses several kinds of C++ parameter passing:

```
int mystery(int a, int* b, int& c) {
    a++;
    (*b)++;
    c++;
    return a;
}

int main() {
    int a = 0;
    int b = 0;
    int c = 0;
    int d = 0;

    mystery(a, &b, c);
    cout << a << " " << b << " " << c << " " << d << endl;

    mystery(c, &d, a);
    cout << a << " " << b << " " << c << " " << d << endl;

    c = mystery(b, &a, d);
    cout << a << " " << b << " " << c << " " << d << endl;

    a = mystery(a, &a, a);
    cout << a << " " << b << " " << c << " " << d << endl;

    return 0;
}
```

6. C++ Classes and Inheritance

Suppose the following C++ classes have been declared:

```
class C1 {
    public:
        void m1() {
            cout << "C1 m1  ";
        }

        void m2() {
            cout << "C1 m2  ";
            m3();
        }

        virtual void m3() {
            cout << "C1 m3  ";
            m1();
        }
};
```

```
class C2 : public C1 {
    public:
        void m1() {
            cout << "C2 m1  ";
        }

        void m2() {
            C1::m3();
            cout << "C2 m2  ";
        }

        virtual void m3() {
            cout << "C2 m3  ";
            m1();
        }
};
```

And suppose a client program declares the following variables:

```
C1* var1 = new C2();

C2* var2 = new C2();
```

Write the output that would be produced by each of the following calls, as it would appear on the console.

`var1->m1();` _____

`var1->m2();` _____

`var1->m3();` _____

`var2->m1();` _____

`var2->m2();` _____

`var2->m3();` _____

7. C++ Programming

Suppose you have a C++ `Date` class with the following members, as was assigned in Homework 7:

- `Date()`
- `Date(`***`year`***`, `***`month`***`, `***`day`***`)`
- `dayOfWeek()`, `dayOfYear()`
- `daysInMonth()`, `daysInYear()`
- `getDay()`, `getMonth()`, `getYear()`
- `isLeapYear()`
- `nextDay()`
- `setDate(`***`year`***`, `***`month`***`, `***`day`***`)`
- `<(`***`date`***`)`, `<=(`***`date`***`)`, `==(`***`date`***`)`, `!=(`***`date`***`)`, `>=(`***`date`***`)`, `>(`***`date`***`)`
- `++`, `--`
- `<< (`***`stream`***`, `***`date`***`)`

Add a method named `daysTo` to the `Date` class, that accepts another date as a parameter and returns an integer representing the number of days you would have to travel in time from this `Date` to reach the given other date.

If this date comes before the parameter date, you should return a positive integer; for example, Sep 19, 2009 is 35 days before Oct 24, 2009. If this date comes after the parameter date, return a negative integer; for example, Sep 19, 2009 would return -499 when compared to May 8, 2008. If the dates are the same, your method should return 0.

The following table shows several `Date` variables and calls to your method and the results that should be returned:

```
Date d1(2009,  9, 19);
Date d2(2009,  9, 29);
Date d3(2009, 10, 24);
Date d4(2008,  5,  8);
Date d5(2009,  9, 19);
Date d6(2008,  1,  1);
```

Call	Return Value
`d1.daysTo(d2)`	10
`d1.daysTo(d3)`	35
`d3.daysTo(d1)`	-35
`d1.daysTo(d4)`	-499
`d1.daysTo(d5)`	0
`d6.daysTo(d4)`	128

It should be possible to call the method repeatedly with the same object and parameter and get the same result (the objects should not be modified as a result of the call). You should accept your date parameter in such a way that no copy of the object is made and such that it is not possible for the client to pass a null date. (You can copy objects in your code if you like, but the process of the client passing its parameter should not cause a copy to be made.)

Unlike in most problems, some aspects of the style of your solution to this problem **will** be graded. In particular, if you write an inelegant solution that repeats several lines of nearly identical code, you may lose a few points. Also, your method should properly use the `const` keyword to indicate what object(s) will not be modified by it.

Write the method declaration that would go in the `Date.h` file, as well as the method definition that would go in the `Date.cpp` file. Place a comment on top of each to indicate which file you want the code to be placed into.

Write your answer on the next page.

---

7. C++ Programming (writing space)