

CSE 333 Midterm Exam
May 9th, 2011

Your Name: _____ **SOLUTION SET** _____

Student ID: _____

General Information:

This is a closed book examination. You have **50 minutes** to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question. There are **10 pages** on this exam (check to make sure you have all of them), and there is a total of **100 points** in all. Write all of your answers directly on this paper. Make your answers as concise as possible. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer.

If you need to tear a page out of the midterm to avoid flipping back and forth, do so carefully; don't tear out any pages with your writing on it, and be careful of the staple!

Our solutions are marked in RED.

	Points available	You earned
Problem 1	25	
Problem 2	24	
Problem 3	25	
Problem 4	25	
Problem 5	1	
Total:	100	

Problem 1: Neutron Dance (25 points)

What is the output of the following C program? (The lines that lead to the printing of output are in bold.) Assume the program is compiled on and runs on a 32-bit CPU. There is space for your answer on the next page.

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

void displayarray(int *x, int len) {
    int i;
    for (i = 0; i < len; i++) {
        if (i == 0)
            printf("%d", x[i]);
        else
            printf(" %d", x[i]);
    }
    printf("\n");
}

int main(int argc, char **argv) {
    int a1[] = {1, 2, 3, 4};
    int a2[4] = {2, 4, 6, 8};
    int *aptr[4] = {a1, &(a2[0])};
    int *p1, *p2, i;

    aptr[2] = (int *) malloc(8*sizeof(int));
    aptr[3] = aptr[2] + 4;

    for (i = 0; i < 4; i++) {
        (aptr[2])[i] = (aptr[0])[i] + (aptr[1])[i];
        (aptr[3])[i] = *(aptr[2]+i) + 1;
    }
    displayarray(aptr[2], 8);

    p1 = *(aptr + 1);
    displayarray(p1, 4);

    p2 = p1 + 2;
    (*p2)++;
    displayarray(p1, 4);

    free(aptr[2]);
    return EXIT_SUCCESS;
}
```

(more space for problem 1.)

3 6 9 12 4 7 10 13

2 4 6 8

2 4 7 8

5 points for “3 6 9 12”

10 points for “4 7 10 13”

5 points for 2 4 6 8

5 points for “2 4 7 8”

Problem 2: Orkin (24 points)

The following program has bugs. Circle the bugs, write code that repairs the bugs, and if your bugfix is on a different line than the bug, write a comment next to the bugfix with the bug line number that it fixes. There is space on the next page for code, but if you use it, indicate at which line number you'd like your code inserted. Ignore stylistic issues; find and fix actual bugs. There are a minimum of 8 changes you need to make. (!)

```
1  #include <stdlib.h>
2  #include <iostream>
3  #include <string>          // needed for std::string
4
5  using namespace std;     // to use "cout" instead of "std::cout", etc.
6  // A class that stores a pair of things of type T.
7
8  template <class T> class Pair {
9  public:
10     Pair() { }           // need default constructor for new Pair<string>[2];
11     Pair(T a, T b): first_(a), second_(b) { }
12
13     void Print() {
14         cout << "(" << first_ << "," << second_ << ")" << endl;
15     }
16
17     void Set(T a, T b) { first_ = a; second_ = b; }
18
19 private:
20
21     T first_, second_;
22 }; // <-- end a class Foo { ... }; with a semicolon
23
24 int main(int argc, char **argv) {
25
26     Pair<string> stringpair("Hello", "World");
27
28     Pair<string> *pairarr = new Pair<string>[2]; // two bugs!
29
30     pairarr[0].first_ = "ham"; // first_, second_ are private!
31
32     pairarr[0].second_ = "cheese"; // ditto
33     Pairarr[0].Set("ham", "cheese"); // use this instead
34     pairarr[1].Set("turtles", "facepaint");
35
36     stringpair.Print();
37
38     pairarr[1]->Print(); // pairarr[1] is not a pointer
39     pairarr[1].Print();
40     delete[] pairarr; // use delete[] to deallocate an array
41
42     return EXIT_SUCCESS;
43 }
44 #include <stdlib.h>
```

(more space for problem 2.)

4 points for each yellow-highlighted section

Problem 3: A frayed knot. (25 points)

In this problem, you will write some C code. Your job is to:

- a) in file “reverse.c”, define a function ReverseString() that:
 - i. accepts a C-style string as a first parameter
 - ii. allocates space for a new string whose length is the same as the first parameter
 - iii. copies the string passed as the first parameter into the newly allocated string, but with the characters in reverse order
 - iv. returns the new string through a second (output) parameter
 - v. returns 0 on failure, 1 on success
- b) in file “reverse.h”, declare a prototype for ReverseString().
- c) in file “test.c”, implement main() to test the correctness of ReverseString().

Be sure to write defensive code. Also, make sure your code has no memory leaks. Feel free to use C standard library functions to get the job done, but #include the right header if you do so. If there are corner cases, do something reasonable to handle them. You have this page, plus two blank pages, to write your code.

(more space for problem 3.)

reverse.h:

```
#ifndef _REVERSE_H_
#define _REVERSE_H_

// This function allocates memory for a string of the same
// length as "in" and then copies "in" to that allocated memory,
// but reverses the characters. The new string is returned
// through output parameter "out". Returns 0 on failure, 1
// on success.
int ReverseString(char *in, char **out);

#endif // _REVERSE_H_
```

reverse.c:

```
#include <stdlib.h> // for malloc()
#include <string.h> // for strlen()
#include <assert.h> // for assert()
#include "reverse.h"

int ReverseString(char *in, char **out) {
    // defensive programming
    assert(in != NULL);
    assert(out != NULL);

    // allocate space for the new string
    int len = strlen(in), i;
    char *ret = (char *) malloc((len+1) * sizeof(char));
    if (ret == 0)
        return 0;

    // copy, but in reverse order
    for (i = 0; i < len; i++)
        ret[(len-1) - i] = in[i];

    // null terminate
    ret[len] = '\0';

    // return our output
    *out = ret;
    return 1;
}
```

(more space for problem 3.)

main.c:

```
#include <stdio.h>
#include <string.h> // for strcmp
#include <stdlib.h> // for free
#include <assert.h> // for assert

#include "reverse.h" // for ReverseString

int main(int argc, char **argv) {
    char *t1 = "hi", *t2 = "123", *t3 = "";
    char *o1, *o2, *o3;

    assert(ReverseString(t1, &o1) == 1);
    printf("%s\n", o1);
    assert(strcmp(o1, "ih") == 0);

    assert(ReverseString(t2, &o2) == 1);
    printf("%s\n", o2);
    assert(strcmp(o2, "321") == 0);

    assert(ReverseString(t3, &o3) == 1);
    assert(o3[0] == '\0');

    free(o1);
    free(o2);
    free(o3);

    return EXIT_SUCCESS;
}
```


Problem 4: En Passant (25 points)

Write the output of the following C++ code.

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

int mystery1(int &a, int *b, int c) {
    a++;
    (*b)--;
    c = a + *b;
    return c;
}

int main(int argc, char **argv) {
    int w = 0, x = 1;
    int &y = x;
    int *z = &x;

    *z = mystery1(w, &x,
                  mystery1(*z, &w, x));

    std::cout << w << " " << x << " " << y << " ";
    std::cout << *z << std::endl;
    return 0;
}
```

0 1 1 1

10 points for the first output

5 points each for the next 3

drawing the box and arrow diagram here helps immensely!

Problem 5: Twisty passages (256 >> 8 points)

The best text-based interactive fiction is clearly:

- a. Zork
- b. Adventure
- c. Dungeon
- d. None of the above
- e. All of the above, including d.
- f. Just give me my free point, please.

We gave you the point for any answer, but especially for e and f. ;)