

Name: _____

**CSE374 Winter 2016, Final Examination
March 17, 2016**

Please do not turn the page until the bell rings.

Rules:

- The exam is closed-book, closed-note, closed-calculator, and closed-electronics.
- **Please stop promptly at 10:20.**
- There are **92 points** total, distributed **unevenly** among **10** questions (many with multiple parts):

Question	Max	Earned
1	9	
2	11	
3	9	
4	14	
5	11	
6	10	
7	8	
8	9	
9	10	
10	1	

Advice:

- Read questions carefully. Understand a question before you start writing.
- **Write down thoughts and intermediate steps so you can get partial credit. But clearly indicate your final answer.**
- The questions are not necessarily in order of difficulty. **Skip around.** Make sure you get to all the problems.
- There is reference material at the end of the exam.
- If you have questions, ask.
- Relax. You are here to learn.

Name: _____

1. (9 points) (Concurrency) Below we give part of the definition of `Queue`, a class representing a queue data structure (first-in, first-out) of `ints`. It uses a linked list internally. The class has a private instance mutex variable that it uses to synchronize concurrent calls to some of its methods.

```
typedef struct node {
    int data;
    struct node *next;
} ListNode;

class Queue{
private:
    ListNode *front;
    ListNode *back;
    mutex mtx;
public:
    int peek() {
        if (front == NULL) {
            throw Exception();
        }
        return front->data;
    }

    int dequeue() {
        // a lock_guard causes the mutex to be held for the duration of the function
        lock_guard<mutex> lock(mtx);
        if (front == NULL) {
            // the lock_guard releases the lock when an exception occurs
            throw Exception();
        }
        int val = front->data;
        front = front->next;
        return val;
        // lock variable goes out of scope, so the mutex is released
    }
};
```

Consider a scenario where we have two threads, $T1$ and $T2$, which share a variable `q`, an instance of the `Queue` class. For each of the following cases where methods of `q` are called simultaneously, describe a bad interleaving or briefly explain why no bad interleaving is possible. We define a bad interleaving to be an interleaving of method operations that causes an incorrect result (wrong value returned, segmentation fault, etc.).

- (a) $T1$ calls `q.peek()` and $T2$ calls `q.peek()`

Name: _____

(b) $T1$ calls `q.peek()` and $T2$ calls `q.dequeue()`

(c) $T1$ calls `q.dequeue()` and $T2$ calls `q.dequeue()`

(d) For each case where you described a bad interleaving, briefly explain how to prevent bad interleavings in that case using mutual exclusion.

Name: _____

2. (11 points) (Concurrency) Below we give part of the definition of `Str`, a class representing a string.

```
class Str {
public:
    int length() const {
        return strlen(st_);
    }

    void append(const Str &s) {
        mtx.lock();
        char *newst = new char[strlen(st_) + strlen(s.st_) + 1];
        strcpy(newst, st_);
        strcat(newst, s.st_);
        delete [] st_;
        st_ = newst;
        mtx.unlock();
    }

private:
    char *st_;
    mutex mtx;
};
```

We are again interested in the consequences of simultaneous calls to certain methods.

- (a) In the case where `append` and `length` are called simultaneously, there is a data race. Briefly explain where and why.

- (b) Briefly explain how to use mutual exclusion to prevent this data race.

Name: _____

- (c) In the case where there are two `Str` instances, `a` and `b`, and a thread $T1$ calls `a.append(b)`, while another thread $T2$ calls `b.append(a)`, there is at least one data race. Briefly explain where and why.

- (d) Explain why the “fix” to `append` given below won't work, and discuss an alternative approach.

```
void append(const Str &s) {
    mtx.lock();
    s.mtx.lock();
    char *newst = new char[strlen(st_) + strlen(s.st_) + 1];
    strcpy(newst, st_);
    strcat(newst, s.st_);
    delete [] st_;
    st_ = newst;
    s.mtx.unlock();
    mtx.unlock();
}
```

Name: _____

3. (9 points) (Integer representation) Please complete the following table of 8-bit integer values (use a two's complement representation for signed values). Powers of two and multiples of 16 are provided as a reference.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0							
128	64	32	16	8	4	2	1							
$16 \times$	2	3	4	5	6	7	8	9	10	11	12	13	14	15
=	32	48	64	80	96	112	128	144	160	176	192	208	224	240

binary	unsigned integer	unsigned hexadecimal integer	signed integer
			-32
		0x3F	
10011100			

Space for scratch work:

Name: _____

4. (14 points) (x86-64 assembly)

(a) For each of the following C snippets, give a series of assembly instructions equivalent to that snippet. Relevant register values are given in each case. You can omit width specifiers on instructions (e.g., `mov` instead of `movq`). Note that register `%rax` is used as the return value, so your assembly code must ensure the return value is stored in `%rax` when the `ret` instruction executes.

- i. C code: `return a * b - c`
a has type `int`
b has type `int`
c has type `int`

Register state:

register	value
<code>%rdi</code>	value of variable a
<code>%rsi</code>	value of variable b
<code>%rdx</code>	value of variable c

x86-64 assembly:

- ii. C code: `return pt->x * pt->x + pt->y * pt->y`
pt has type `Point*`
Point is a typedef for a structure representing a 2D point:

```
typedef struct {  
    int x;  
    int y;  
} Point;
```

Register state:

register	value
<code>%rdi</code>	value of pt

x86-64 assembly:

Name: _____

- (b) For the following series of assembly instructions, give the equivalent snippet of C. Relevant register values and variable names are given.

x86-64 assembly:

```
mov (%rdi,%rsi,4), %rbx
mov 4(%rdi,%rsi,4), %rcx
mov %rcx, (%rdi,%rsi,4)
mov %rbx, 4(%rdi,%rsi,4)
```

Register state:

register	value
%rdi	value of variable <code>array</code> that has type <code>int*</code>
%rsi	value of variable <code>index</code> that has type <code>int</code>

C code:

Name: _____

5. (11 points) (Bash scripting) Write a shell script `projectsize` that prints the total size in bytes of the C source files in a directory. The command

```
projectsize DIR
```

should print the total size in bytes of the C source files in the directory `DIR`.

Your script should meet the following specification:

1. Print an informative error message to `stderr` if `DIR` does not exist.
2. If no argument is given, print the total size in bytes of the C source files in the **current** directory
3. Treat as a C source file any file that ends in `.c` or `.h`.
4. Ignore subdirectories of `DIR` (or the current directory).

You can use `wc` to get the number of bytes in a file. To test if a file ends in `.c` or `.h`, you can use the `extension-test` command. `man extension-test` provides the following documentation:

NAME

```
extension-test -- test if a file ends in a specific extension
```

SYNOPSIS

```
extension-test [OPTION]... FILE EXT
```

DESCRIPTION

The `extension-test` utility checks if `FILE` ends in `EXT`, and sets its exit status accordingly (similar to `[]` command).

`EXT` should not include the `'.'` part of the extension.

The following options are available:

```
-i
```

```
    Ignores case when comparing FILE and EXT
```

```
--mult
```

```
    Multiple EXT are given, exit with success if FILE ends in any of the given EXT
```

```
-x
```

```
    In addition to checking that FILE ends in EXT, also checks that FILE is executable
```

Please write your script on the following page.

Name: _____

Write your answer to problem 5 here.

Name: _____

6. (10 points) (C programming) Your task is to implement the C standard library function `strcat`. You may implement any helper functions you find useful. Your `strcat` function must meet the following specification:
1. Matches the declaration `char * strcat (char * destination, char * source)`
 2. Assumes `destination` and `source` are valid C strings (i.e., non-NULL, end in null terminator).
 3. Appends a copy of the `source` string to the `destination` string. The terminating null character in `destination` is overwritten by the first character of `source`, and a null-character is included at the end of the new string formed by the concatenation of both in `destination`.
 4. Assumes `destination` is large enough to contain the concatenated resulting string.
 5. Returns `destination`
 6. Does not use any standard library functions (e.g., `strlen`)

Name: _____

7. (8 points) (Toolchain) Recall that there are several steps needed to build an executable program from source files and libraries. Below is a list of several possible errors that can occur when a program is compiled, linked, or executed. For each error, indicate the earliest stage in the process of building and executing the program where it is **always possible** to discover the error and produce some sort of error message or failure. (Note, for example, that some errors can be detected early, say division by 0 if the program contains `x/0` in the source code, but in general division by 0 cant be detected until the program is executed if it is dividing `x/y` and the value of `y` is not known until runtime.)

Identify where (when) each possible error can definitely be detected. Fill in one of the following codes in the space provided:

- *cpp* - C preprocessor
- *comp* - C compiler
- *ld* - linking/loading step
- *exe* - during program execution
- *can't* - cannot be detected always (including illegal programs that might not actually fail during execution)

_____ calling `free` on a pointer to stack-allocated data

_____ dereferencing a pointer `ptr`, where `ptr == 0x0`

_____ using a variable before it has been declared

_____ making a typo when including a standard library header (e.g., `#include <sgtio.h>`)

_____ a function returns a pointer to one of its local variables

_____ when the following code is compiled and/or run (the code below is **all** the code there is)

```
int Foo();

int main() {
    int a = Foo();
    return 0;
}
```

_____ the function `char* lookup(ListNode *list, char *word)` is called with the parameters in the wrong order

_____ `strncpy` is used to copy a `char*`, but the null terminator is not copied

Name: _____

8. (9 points) (C programming) You are working on a C data structure for a doubly-linked list (i.e., a linked list that has both next and previous links for each node). Each node stores a **struct** representing a user. Here are the **structs** you're using:

```
typedef struct {
    int id;
    char *name;
} User;

typedef struct node {
    User *user;
    struct node *prev;
    struct node *next;
} ListNode;
```

The nodes and user data they store are allocated on the heap. Your task is to implement a function to remove a node from the list. Your function must meet the following specification:

1. Matches the declaration `int remove(ListNode *root, int target_id)`
2. Removes the first node after `root` that has a `user` with an `id` equal to `target_id` (`root` itself will never be removed)
3. Deallocates all memory used by the removed node and returns `true`.
4. If there are no nodes in the list, or no node has an `id` equal to `target_id`, returns `false`.

Name: _____

9. (10 points) (C++) Consider the following program, which compiles and executes without errors.

```
class Rational {
public:
    Rational();
    Rational(int n);
    Rational(int n, int d);
    Rational(const Rational &other);
    ~Rational();
    Rational &operator=(const Rational &other);
private:
    int num;
    int denom;
};
// implementations
Rational::Rational(): num(0), denom(0)
    { cout << "default constructor" << endl; }
Rational::Rational(int n): num(n), denom(0)
    { cout << "one int constructor" << endl; }
Rational::Rational(int n, int d): num(n), denom(d)
    { cout << "two int constructor" << endl; }
Rational::Rational(const Rational &other): num(other.num), denom(other.denom)
    { cout << "copy constructor" << endl; }
Rational::~~Rational() { cout << "destructor" << endl; }
Rational & Rational::operator=(const Rational &other) {
    cout << "assignment" << endl;
    this->num = other.num;
    this->denom = other.denom;
    return *this;
}
int main() {
    Rational *r2 = new Rational(1, 2);
    Rational r1();
    Rational *r3 = &r1;
    r1 = *r2;
    Rational r4 = 5;
    *r2 = r4;
    delete r2;
    Rational r5 = r1;
    return 0;
}
```

What output is produced when this program is executed?

Name: _____

10. (1 point) (Haiku) Please compose a haiku about something you learned this quarter. Here's mine:

inheritance can
cause so many bugs, you are
better off without

Name: _____

Reference

This is an incomplete list. **Just because a command or option is documented here doesn't mean there is a question that uses it.**

Bash

`wc [OPTION]... [FILE]...`

Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified.

`-c, --bytes`
print the byte counts
`-l, --lines`
print the newline counts
`-w, --words`
print the words counts

shell scripting

`$(cmd)` substitute with the stdout from running cmd
`$((expr))` substitute with the result of evaluating expr -- useful for math
`$n` nth argument (\$0 is the command itself)
 `$#` number of arguments (does not include \$0)
 `$@` a list of all the arguments (does not include \$0)
 `$?` the exit status of the most recent command
 `shift` discard the first argument (\$1) and move the remaining arguments down (\$2 becomes \$1 and so on, this affects \$# and \$0).

```
for item in list_of_things
do
...
done
```

```
if TEST
then
...
fi
```

tests:

```
[ -d file ] true if file exists and is a directory
[ -e file ] true if file exists, regardless of type
[ -f file ] true if file exists, and is a regular file
[ -n string ] true if length of string is nonzero
[ -z string ] true if length of string is zero
[ s1 = s2 ] true if the strings s1 and s2 are identical.
[ s1 != s2 ] true if the strings s1 and s2 are not identical.
[ n1 -eq n2 ] true if integer n1 is equal to integer n2.
similarly for not equal (-ne), greater than (-gt),
and less than (-lt)
```


Name: _____

x86-64 assembly

instructions:

instruction	effect	description
mov S, D	$D \leftarrow S$	move
add S, D	$D \leftarrow D + S$	add
sub S, D	$D \leftarrow D - S$	subtract
imul S, D	$D \leftarrow D * S$	multiply
ret	returns, uses value in <code>%rax</code> as return value	return

operand forms:

type	form	operand value
immediate	$\$D$	D
register	$\%R$	value stored in register R
memory	D	value at memory location D
memory	$(\%R)$	value at memory location given by value stored in register R
memory	$D(\%R_b)$	value at memory location $D + R_b$
memory	$(\%R_b, \%R_i)$	value at memory location $R_b + R_i$
memory	$D(\%R_b, \%R_i)$	value at memory location $D + R_b + R_i$
memory	$(, \%R_i, s)$	value at memory location $R_i \cdot s$
memory	$D(, \%R_i, s)$	value at memory location $D + R_i \cdot s$
memory	$(\%R_b, \%R_i, s)$	value at memory location $R_b + R_i \cdot s$
memory	$D(\%R_b, \%R_i, s)$	value at memory location $D + R_b + R_i \cdot s$

The scaling factor s must be either 1, 2, 4, or 8.