CSE 333 Midterm Exam July 24, 2017

Name _____ UW ID# _____

There are 6 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, closed electronics, closed telepathy, open mind.

If you don't remember the exact syntax for something, make the best attempt you can. We will make allowances when grading.

Don't be alarmed if there seems to be more space than is needed for your answers – we tried to include more than enough blank space.

Relax, you are here to learn.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 14

2. _____ / 16

3. _____ / 20

4. _____ / 16

5. _____ / 32

6. _____ / 2

Question 1. (14 points) Making things. Suppose we have the following C header and implementation files, showing the various `#include` and `#if…` directives in each one:

```
======
widget.h
======
#ifndef _WIDGET_H_
#define _WIDGET_H_
#include "thing.h"
...
#endif

======
widget.c
======
#include "widget.h"
...

======
usestuff.c
======
#include "widget.h"
#include "thing.h"
...
```

```
======
thing.h
======
#ifndef _THING_H_
#define _THING_H_
...
#endif

======
thing.c
======
#include "thing.h"
...
```

We've been building the program with the following `gcc` command:

```
gcc -Wall -g -std=c11 -o usestuff *.c
```

One of the new interns says that we should update our infrastructure and use `make` to build the program. The intern has written the `Makefile` on the next page, which is supposed to do the job. Unfortunately it doesn't work right.

Your job is to fix the `Makefile` so that it works properly. It should produce a result that is equivalent to the one produced by the `gcc` command above, including using the same compiler options (`-Wall`, `-g`, `-std=c11`), but, of course, skipping any compile or link steps that are not needed. In case it matters, the `main` function is in file `usestuff.c`. You may assume that each of the commands in the current `Makefile` begins correctly with a tab character – that's not the problem.

Hint: the `gcc` option `-c` will compile a source file and stop after producing the corresponding `.o` file.

Write your correction on the `Makefile` on the next page. You may remove this page from the exam if you wish.

Question 1. (cont.) Make corrections to the following `Makefile` so it works as specified on the previous page.

```
widget.o: widget.c
     gcc -Wall -g -std=c11 widget.c

thing.o: thing.c
     gcc -Wall -g -std=c11 thing.c

usestuff.o: usestuff.c
     gcc -Wall -g -std=c11 usestuff.c

usestuff: widget.c thing.c usestuff.c
     gcc -Wall -g -std=c11 \
          -o usestuff widget.o thing.o usestuff.o
```

(Note: the '\' at the end of the last `gcc` command was used to break a single long command into two lines so it would fit on the page. The effect is exactly the same as if the entire command had been written on a single, long line. You can retain the '\' in your answer or write the code on a single line if appropriate.)

Question 2. (16 points) Preprocessor. Suppose we have the following two files:

prepro.h:

```
#ifdef MAGIC
#define MAX(x, y) 42
#else
#define MAX(x, y) x > y ? x : y
#endif
#define ADD(x, y) x + y
#define XVII 17
#define CSE 333
```

prepro.c:

```
#include <stdio.h>
#include "prepro.h"
int main() {
  printf("%d\n", 1 + MAX(XVII, CSE) );
  printf("%d\n", ADD(XVII, 1) * 2);
  return 0;
}
```

(a) (8 points) Show the result produced by the C preprocessor when it processes file `prepro.c` (i.e., if we were compiling this file, what output would the preprocessor send to the C compiler that actually translates the program to machine code?). You should ignore the `#include <stdio.h>` directive since that includes library declarations that we do not have access to. Write the rest of the preprocessor output below.

(b) (4 points) What output does this program print when it is executed? (It does compile and execute without errors.)

(c) (4 points) Now, suppose we add the option `-DMAGIC` to the `gcc` command when we compile the program. This causes the program to be compiled as if `#define MAGIC` had been added at the beginning of the file. How does this change your answers to parts (a) and (b)?

Question 3. (20 points) Some simple C programming. Write a program on the next page that works as follows: The program has a single command-line argument giving the name of a text file. The program should open that file and copy its contents to `stdout`, except that any input line that contains more than 40 characters should be truncated to 40 characters (not counting the `\n` newline at the end of the line). For example, if the input file `alice.txt` contains:

```
"Who are YOU?" said the Caterpillar.
This was not an encouraging opening for a conversation.
Alice replied, rather shyly, "I-I hardly know, sir, ...
```

then the output of the program should be as follows, truncated to 40 characters:

```
"Who are YOU?" said the Caterpillar.
This was not an encouraging opening for
Alice replied, rather shyly, "I-I hardly
```

You must read input lines by calling a function to read each complete line, not by reading one character at a time. The program should terminate with `EXIT_SUCCESS` if all is well, or print an appropriate (brief) message and terminate with `EXIT_FAILURE` if problems occur. You may assume:

- The file contains only ASCII text and each line has a `\n` at the end
- No line contains more than 500 characters, including any `\n` or `\0` at the end
- It's fine to use standard C library routines for file I/O and string handling
- Any necessary headers have already been `#include`d for you
- You can put all the code in a single `main` function if that makes sense

Hints: `fgets`, `printf`, `strlen`, etc.

A bit of reference information:
Some useful string functions, if you need them. All string arguments have type char*.

- strlen(s) # of characters (bytes) in s, not including the '\0' byte at the end.
- strcpy(dst,src) copies src to dst; also strncpy(dst,src,max)
- strcat(dst,str) appends a copy of src to the end of dst, also strncat(dst,src,max)
- strcmp(x,y) returns 0 if strings x and y are the same, <0 if x<y, and >0 if x>y.

A few stdio function prototypes and hints

- fopen(filename, "rb") – open to read bytes
- fgets(buffer, max, FILE* f) – returns NULL if eof or error, otherwise reads a line of up to max-1 characters into buffer, including the \n, and adds a \0 at the end
- fread(buf, 1, count, FILE* f)
- fwrite(buf, 1, count, FILE* f)
- fprintf(format_string, data…, FILE *f)
- feof(FILE* f) – is eof indicator for f set?
- ferror(FILE* f) – was there an error on f?

(Continued on the next page. You can remove this page from the exam if you wish.)

Question 3. (cont.) Write your answer to the question below. Some #includes and the header for main are given to save some writing:

```
#include <stdio.h>   // printf, scanf, fopen, etc.
#include <string.h> // string library
#include <stdlib.h>  // EXIT_SUCCESS, EXIT_FAILURE
// add any additional preprocessor commands you need below




int main(int argc, char ** argv) {









}
```

Question 4. (16 points) More C programming. This question involves writing a function that uses the HW1 linked lists. A copy of the `LinkedList.h` header file, which is the only one that might be needed for this question, is provided on separate pages. You should assume that this header, and any other necessary header files and declarations, have already been `#include`d and you do not need to write those in your answer.

One of our clients is using our `LinkedList` package to keep track of lists of integers. Each item in a list is a pointer to an `int` variable allocated on the heap. The code the client is using to store new values in a list looks like roughly like this (error checks omitted):

```
bool insert(LinkedList ll, int num){
    int *tmp = (int *)malloc(sizeof(int));
    *tmp = num;
    PushLinkedList(ll, tmp))
    ...
}
```

The client would like to implement a function to count the number of occurrences of a given integer value in a list, and, since you're an expert, you've been asked to do the job. Here is the prototype and specification for this new function:

```
// Return the number of occurrences of a value in a list.
//
// Arguments:
//
// - ll: the list to examine
//
// - num: the number to search for
//
// Returns the number of occurrences of num in list ll,
// or return -1 if an error occurs.
int LLLookupNum(LinkedList ll, int num);
```

Complete the implementation of this function on the next page. For full credit, your solution must use functions in the `LinkedList` package when appropriate and not duplicate code that is already implemented elsewhere. (Specifically, you should use an iterator and related functions to process the list.) Remember that this is a client function, not a part of the `LinkedList` package.

Hint: Don't panic! The solution isn't particularly long – the sample solution is about a dozen lines of code. Of course, yours isn't required to be any particular length.

You may remove this page from the exam if you wish.

Question 4. (cont.) Complete the implementation of `LLLookupNum` below.

```
#include "LinkedList.h"
// other headers not shown to save space

int LLLookupNum(LinkedList ll, int num) {

  // Add your code below









}
```

Question 5. (32 points) Some questions about `Str`ings. A long question with probably too many parts.

Recall the small C++ string class from lecture:

```
class Str {
 public:
  Str();                // default constructor
  Str(const char *s);   // c-string constructor
  Str(const Str &s);    // copy constructor
  ~Str();               // destructor

  // operations
  int length() const;             // = length of this string
  char * c_str() const;           // = new char* copy of this string
  void append(const Str &s);      // append s to this
  Str &operator=(const Str &s);   // assignment

  // stream output
  friend std::ostream &operator<<(std::ostream &out, const Str &s);

 private:
  char *st_;  // c-string on heap with data bytes terminated by \0
};
```

Answer the following question about this class and its implementation. For the first few parts, you are asked to say whether a proposed change in the code is correct, fails to compile properly, compiles but does not execute as it is supposed to, or compiles and executes properly. You also need to give a short explanation for your answer.

(a) (5 points) The copy constructor has a parameter with type `const Str &` (i.e., constant reference to `Str`). Suppose we delete the reference `(&)` and change this constructor and its implementation to have the following parameter declaration:

```
  Str(const Str s);   // copy constructor
```

Will the class compile without errors with this change? (circle one)

Yes No Maybe

If it does compile, will it work correctly when code that uses it is executed (circle one)?

Yes No Maybe Doesn't apply – won't compile

Give a brief (1-2 sentence) specific, technical explanation for your answers:

(continued next page)

Question 5. (cont.) (b) (5 points) This class includes an assignment operator with the following implementation:

```
Str &Str::operator=(const Str &s) {
  if (this == &s) {
    return *this;
  }
  delete [] st_;
  st_ = new char[strlen(s.st_)+1];
  strcpy(st_, s.st_);
  return *this;
}
```

Suppose we change the return type of this function to be `Str` instead of `Str&` (i.e., delete the first `&` to get `Str Str::operator=(const Str &s)`). Will the class compile without errors with this change? (circle one)

Yes No Maybe

If it does compile, will it work correctly when code that uses it is executed (circle one)?

Yes No Maybe Doesn't apply – won't compile

Give a brief (1-2 sentence) specific, technical explanation for your answers:

(continued next page)

Question 5. (cont.) (c) (5 points) This class has an associated stream output operation with the following implementation:

```
ostream &operator<<(ostream &out, const Str &s) {
  out << s.st_;
  return out;
}
```

Suppose we change the return type of this function by deleting the `&` so the function heading is as follows:

```
ostream operator<<(ostream &out, const Str &s) {
```

Will the class compile without errors with this change? (circle one)

Yes No Maybe

If it does compile, will it work correctly when code that uses it is executed (circle one)?

Yes No Maybe Doesn't apply – won't compile

Give a brief (1-2 sentence) specific, technical explanation for your answers:

(continued on next page)

Question 5. (cont.) (d) (12 points) The following program prints "hello cse333" in a complicated way using a linked list of `Str`s. (It does compile and run successfully.)

```
struct SNode { // A linked list node for Str values
  Str value_;
  SNode *next_;
  // convenience constructor
  SNode(Str s, SNode *p) : value_(s), next_(p) { }
};

int main() {
  Str s1("cse333");
  SNode n1(s1, nullptr);
  Str * s2 = new Str("hello");
  SNode *n2 = new SNode(*s2, &n1);
  //// HERE ////
  cout << n2->value_ << " " << n2->next_->value_ << endl;
  return 0;
}
```

Below, draw a memory diagram showing *exactly* how memory is laid out when execution reaches the point labeled `//// HERE ////` in the main program. Be sure to clearly show what variables and objects are allocated on the stack, which ones are on the heap, and how they are connected by pointers. You should omit any temporary objects that might have been created by the compiler.

Stack memory (belonging to `main`)	Heap memory

Question 5. (concluded!) (e) (5 points) When we ran the program from the previous part of the question using `valgrind` it reported that there were memory leaks:

```
==5180== HEAP SUMMARY:
==5180==     in use at exit: 36 bytes in 4 blocks
==5180==   total heap usage: 8 allocs, 4 frees, 63 bytes allocated
==5180==
==5180== LEAK SUMMARY:
==5180==    definitely lost: 24 bytes in 2 blocks
==5180==    indirectly lost: 12 bytes in 2 blocks
==5180==      possibly lost: 0 bytes in 0 blocks
==5180==    still reachable: 0 bytes in 0 blocks
==5180==         suppressed: 0 bytes in 0 blocks
```

Here is another copy of the code. Indicate on the code the changes that should be made to fix the program so it does not leak memory. You should not use smart pointers (i.e., no `unique_ptrs`). Just add the needed code to fix the memory leaks.

```
struct SNode { // A linked list node for Str

  Str value_;

  SNode *next_;

  // convenience constructor

  SNode(Str s, SNode *p) : value_(s), next_(p) { }

};

int main() {

  Str s1("cse333");

  SNode n1(s1, nullptr);

  Str * s2 = new Str("hello");

  SNode *n2 = new SNode(*s2, &n1);

  //// HERE ////

  cout << n2->value_ << " " << n2->next_->value_ << endl;

  return 0;

}
```

Question 6. (2 free points) (All reasonable answers receive the points. All answers are reasonable as long as there is an answer. ☺)

(a) (1 point) What question were you expecting to appear on this exam that wasn't included?

(b) (1 point) Should we include that question on the final exam? (circle or fill in)

Yes

No

Heck No!!

$!@$^*% No !!!!!

Covfefe

None of the above. My answer is _____.