CSE 374 Final Exam 12/14/15

Name ________________________________ UW ID# _________________

There are 10 questions worth a total of 110 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 laptops, closed twitter, closed telepathy, closed Facebook, etc.

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

Score _________________ / 110

1. ______ / 10

2. ______ / 13

3. ______ / 20

4. ______ / 9

5. ______ / 8

6. ______ / 10

7. ______ / 10

8. ______ / 8

9. ______ / 10

10. ______ / 12

Question 1. (10 points, 1 each) 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 can't 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)

_____ Function is called but is not defined anywhere

_____ File named in `fopen` call does not exist

_____ Variable declared more than once in a function

_____ File `foo.h` in `#include "foo.h"` does not exist

_____ Function called with wrong number of parameters

_____ In an array reference `a[i]`, the value of `i` exceeds the length of the array

_____ Same function is defined with external linkage in more than one C file

_____ In an assignment statement `x=e;` the type of expression `e` is not assignable to a variable with the type of variable `x`.

_____ In an assignment statement `x=e; x` does not denote an lvalue expression

_____ In an expression `p->x`, pointer `p` is `NULL`

Question 2. (13 points) Consider the following two header files and main program:

File header.h:

```
#ifndef HEADER_H
#define HEADER_H
#include <stdio.h>

#define A(x,y) x+y

#ifdef HOLIDAY
#define PR(s) printf("%s %s %s\n", s, s, s);
#else
#define PR(s) printf("%s%s\n", s, s);
#endif
#endif  // HEADER_H
```

File prepro.c:

```
#define HOLIDAY

#include "header.h"
#include "footer.h"
#include <stdio.h>

int main() {
  printf("MAGIC = %d\n", MAGIC);
  printf("%d\n", 2*A(3,4));
  PR("ho");
  return 0;
}
```

File footer.h:

```
#ifndef FOOTER_H
#define FOOTER_H

#define MAGIC 17

#endif  // FOOTER_H
```

(a) (8 points) Below write the output produced by cpp (the preprocessor) when it processes file `prepro.c`. This is the output sent from cpp to the C compiler.

(b) (5 points) What output is produced by this program when it is compiled and executed?

Question 3. (20 points) A little C programming. The following `struct` can be used to define the nodes in a linked list of integer values:

```
struct node {
  int val;                // integer value in this node
  struct node * next;     // next node or NULL if none
};
```

For this problem, give an implementation of function `clone` below that will return a pointer to an exact copy of the linked list that is its argument. The nodes of the new copy of the list should all be allocated on the heap by `malloc` as needed. (Hint: you may not need nearly this much space. You may define auxiliary functions if that is useful and makes your solution clearer or easier to understand.)

```
// return a newly allocated clone (copy) of the
// linked list with initial node p
struct node * clone(struct node *p) {







}
```

Question 4. (9 points) Another of those questions that won't go away. Here's a C program that involves pointers and structs. What output is produced when this program is executed (and it does compile and execute without errors). The first line of output is provided for you and blanks are provided for the remaining numbers. Hint: struct parameters work like all other types (except arrays) – e.g., call by value, or use a pointer if the function is to manipulate the original variable in the caller.

```
struct s {
  int a;
  int b;
  int c;
};

void f(struct s* s) {
  s->a = 2 * s->b;
  s->c = s->a - s->c;
}

void g(struct s s1, struct s* s2) {
  s1.a = 2 * s2->b;
  s2->c = s1.a - s2->c;
}

void h(struct s s) {
  s.a = 2 * s.b;
  s.c = s.a - s.c;
}

void print(struct s s) {
  printf("s: a= %d  b= %d  c= %d\n", s.a, s.b, s.c);
}

int main(void) {
  struct s s = { 1, 2, 3 };
  print(s);
  f(&s);
  print(s);
  g(s, &s);
  print(s);
  h(s);
  print(s);
}
```

Output:

s: a = 1 b = 2 c = 3

s: a = ______ b = ______ c = ______

s: a = ______ b = ______ c = ______

s: a = ______ b = ______ c = ______

Question 5. (8 points) A little debugging. Consider the following C program (`#includes` omitted to save space):

```
void stringcpy(char *dst, char *src) {
  int len = strlen(src);
  for (int i = 0; i < len; i++) {
    dst[i] = src[i];
  }
}

int main() {
  char *str = "hello cse374";
  char *cpy = (char *) malloc(13);  // enough for "hello cse374"
  stringcpy(cpy, str);
  assert(strlen(str) == strlen(cpy));
  printf("%s\n%s\n", str, cpy);
  free(cpy);
  return 0;
}
```

The program compiles and executes as expected without any apparent errors or assertion failures. But when we run it under valgrind, we get the following messages:

```
==1744== Conditional jump or move depends on uninitialized
         value(s)
==1744==    at 0x4C2BC38: strlen
       (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
==1744==    by 0x400708: main (stringcpy.c:17)
```

What does this message mean? What is the error and how should it be fixed? (Be brief!)

Question 6. (10 points) A little more debugging, C++ this time. Consider the following C++ program.

```
int* f(int* n) {
  int* a = new int[*n];
  for (int i = 0; i < *n; ++i)
    a[i] = i;
  return a;
}

int* g(int* n) {
  *n = 5;
  return n;
}

int main(void) {
  int* a = new int;
  *a = 3;
  int* b = f(a);
  int* c = g(b);

  delete a;
  delete b;
  delete c;
}
```

This program has memory management problems. On some systems it manages to terminate without any error messages; on others it crashes or produces error messages. Even if it terminates without crashing, `valgrind` reports additional errors.

What's wrong and how should it be fixed? Either explain the problems and solutions clearly below, or indicate the answers on the code above. Either way, be sure your answer is clear and easy to understand.

Question 7. (10 points) Tools: make. We have a small program directory containing the following three files: c1.c, c2.c, and hdr.h. The header file hdr.h is #included by both C files (c1.c and c2.c). While it is possible to compile the program with the command

```
   gcc -Wall -g -std=c11 -o prog c1.c c2.c
```

we'd like to do better than that.

Below write the contents of a Makefile that will produce the same executable program as the above gcc command when a make command is entered, but will only recompile source files when needed and will save the old .o files to reuse later if they don't need to be recompiled.

You also should include a "clean" target so that make clean will remove any files built by running make previously.

Hint: gcc -c

Question 8. (8 points) Tools: git. Now that you've learned how to use git in cse 374 you've started using it on all of you projects. You've been working with a partner on a program to control spaceships and you've just added a `launch()` function to the code. You've tested your code and everything works just fine.

So you tell your partner to use `git pull` to get your latest changes and try out the `launch` function. She runs these commands

```
git pull
make
```

and gets this error message:

```
In function `main': launch-control.c:134:
                    undefined reference to `launch'
collect2: error: ld returned 1 exit status
```

(a) (3 points) What is the most likely problem here? After all, it works fine when you run it on your computer.

(b) (5 points) What commands (particularly git) should you and/or your partner enter to fix the problem? When you're done, both of you should have cleanly-compiled, up-to-date copies of the program.

Question 9. (10 points) A simple(??) C++ class. Consider the following program, which compiles and executes without errors.

```
// A class that contains an integer - much like Java’s Integer
class Integer {
 public:
  Integer();
  Integer(int n);
  Integer(const Integer &other);
  ~Integer();
  Integer &operator=(const Integer &other);
 private:
  int val;  // the int inside this Integer
};

// implementations
Integer::Integer(): val(0)
                    { cout << "default constructor" << endl; }
Integer::Integer(int n): val(n)
                    { cout << "int constructor" << endl; }
Integer::Integer(const Integer &other): val(other.val)
                    { cout << "copy constructor" << endl; }
Integer::~Integer() { cout << "destructor" << endl; }

Integer & Integer::operator=(const Integer &other) {
  cout << "assignment" << endl;
  this->val = other.val;
  return *this;
}

int main() {
  Integer n1;
  Integer n2 = 17;
  Integer n3 = n2;
  n1 = n3;
  n3 = 42;
  return 0;
}
```

What output is produced when this program is executed?

Question 10. (12 points) The return of the unavoidable ~~dreaded traditional annoying exasperating expected~~ C++ "what does this print" question. What output is produced when the following program is executed? (It does compile and execute without errors.)

```
#include <iostream>
using namespace std;

class A {
 public:
          void m1() { m2(); cout << "A::m1" << endl; }
          void m2() { cout << "A::m2" << endl; }
  virtual void m3() { cout << "A::m3" << endl; }
};

class B: public A {
 public:
          void m1() { m2(); cout << "B::m1" << endl; }
  virtual void m2() {       cout << "B::m2" << endl; }
};

class C: public B {
 public:
  virtual void m2() { cout << "C::m2" << endl; }
  virtual void m3() { cout << "C::m3" << endl; }
};

int main() {
  A* a = new B();
  a->m1();
  a->m2();
  cout << "-----" << endl;
  B* b = (B*)a;
  b->m1();
  b->m2();
  cout << "-----" << endl;
  B* bc = (B*)new C();
  bc->m1();
  bc->m3();
  return 0;
}
```

Output:

Best wishes for the holidays and the New Year! The CSE 374 Staff