# CSE 333 19wi Midterm Exam Feb. 15, 2019 Sample Solution

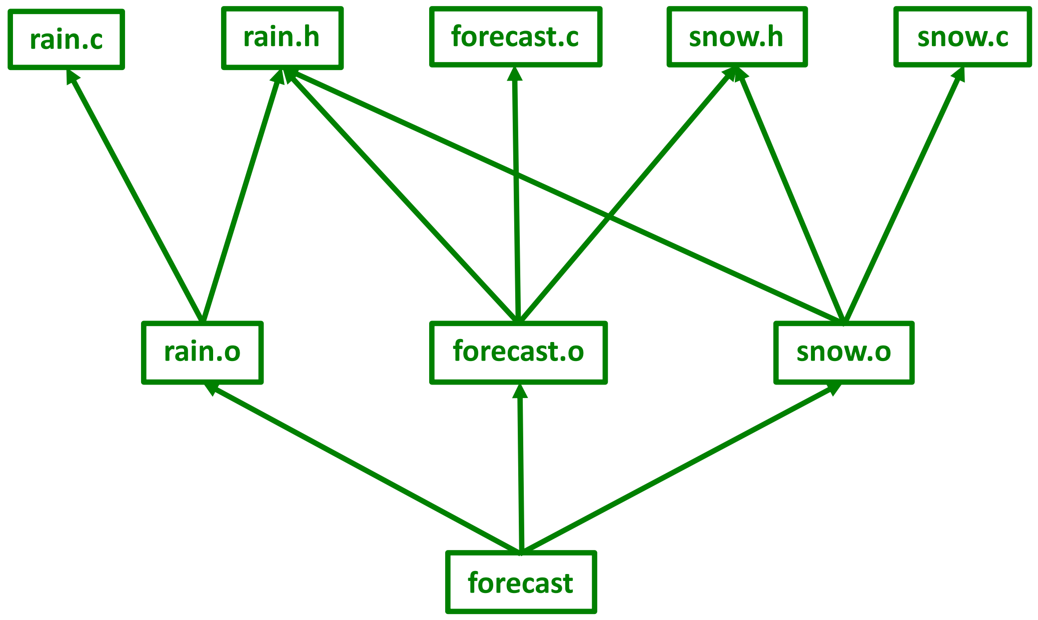
**Question 1.** (16 points) Build tools and `make`. Suppose we have the following `Makefile` in the current directory.

**(Changes from part (c) shown in bold green below)**

```
rain.o: rain.h rain.c wind.h
      gcc -Wall -g -c rain.c
snow.o: snow.h snow.c rain.h
      gcc -Wall -g -c snow.c
forecast: forecast.o rain.o snow.o wind.o
      gcc -Wall -g -o forecast forecast.o rain.o snow.o wind.o
forecast.o: forecast.c rain.h snow.h wind.h
      gcc -Wall -g -c forecast.c
wind.o: wind.c wind.h
      gcc -Wall -g -c -wind.c
```

(a) (6 points) In the space below, draw a dependency graph (diagram) showing the dependencies between the files as specified by the above Makefile. Your drawing should have an arrow from each file to the files that it depends on, as in the diagram to the right, where `foo` depends on `fum`, which depends in turn on `fie`. Hint: if the `gcc -c` option is present but no `-o` option is given, the default output file name for `gcc -c xyz.c` is `xyz.o`.

$$\begin{tikzcd}
\texttt{fie} \\
\texttt{fum} \arrow[u] \\
\texttt{foo} \arrow[u]
\end{tikzcd}$$



(continued on next page)

**Question 1. (cont.)** (b) (3 points) If we just type "`make`" in the directory containing this `Makefile`, what happens? Assume that all of the source files (`.c` and `.h`) are present in the directory along with the `Makefile`, but that none of the other files (`.o`, etc.) named in the `Makefile` are present when we run make.

**`gcc` will be executed to compile `rain.c` to produce `rain.o` and then `make` will stop. The default operation of `make` with no parameters is to process the first target found in the file.**

**(For the `Makefile` to do what the user probably intended, which is to build the complete program by default, the `forecast` target should be moved to be first in the `Makefile`, or else a rule like `all: forecast` should be added at the beginning.)**

------------------

Now suppose we've been working on this project and want to add a new source file `wind.c` along with an associated header file `wind.h`. File `wind.c` only includes the `wind.h` header; it does not use any of the others. Files `forecast.c` and `rain.c` have been modified to use the new functions defined in these new files, with the appropriate `#includes` and calls to the new functions.

(c) (4 points) On the previous page, write in the modifications needed to the `Makefile` so these new files are included in the program and are correctly compiled (and recompiled) as needed when the program is built.

**(See changes on previous page)**

(c) (3 points) Once the new `wind.h` and `wind.c` files have been tested and the `Makefile` updated in our working copy of the program, we need to be sure to update the `git` repository. Write the necessary `git` commands below to update the repository to reflect these additions and changes.

```
git add wind.h wind.c Makefile
git commit -m "add wind files to project"
git push
```

**(Of course there are other `git` commands that would have the same effect, but these are sufficient. The `Makefile` does need to be included since it was modified as part of the changes. The question should have been more specific that the commands should update the remote repository as well as the local one, but everyone included the `push` so that turned out not to be an issue. It also would be a good idea to do a `git pull` before `commit` to minimize the chances of a merge conflict, but that was not required as part of the answer.)**

**Question 2.** (12 points) Preprocessor – more madness than usual. We have the following three C files written by someone who clearly doesn't quite understand things:

foo.h:

```
#ifndef _FOO_H_
#define _FOO_H_
int THE_ANSWER = 42;
#define BAR(x) x * THE_ANSWER
int bar(int);
#endif  // _FOO_H_
```

foo.c:

```
#include "foo.h"
int bar(int x) {
  return x * THE_ANSWER;
}
```

main.c:

```
#include <stdio.h>
#include "foo.c"  // ☜☜ !!!!
int main(int argc,
         char **argv) {
  printf("%d\n", bar(2));
  printf("%d\n", BAR(1 + 1));
  return 0;
}
```

(a) (9 points) Show the result produced by the C/C++ preprocessor when it processes file `main.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.

```
int THE_ANSWER = 42;
int bar(int);
int bar(int x) {
  return x * THE_ANSWER;
}
int main(int argc, char **argv) {
  printf("%d\n", bar(2));
  printf("%d\n", 1 + 1 * THE_ANSWER);
  return 0;
}
```

(b) (3 points) What happens when we try to compile and execute `main.c` by itself using `gcc -Wall -g -std=c11 -o main main.c` (i.e., does it compile, and, if so, what happens when it runs, etc.)? If there is an error, explain the problem briefly.

**The program compiles and executes successfully and produces this output:**

```
84
43
```

**Question 3.** (15 points) Nodes in a binary search tree of C strings can be defined as follows:

```
typedef struct bst_node {
  char *str;                // heap-allocated string for this node
  struct bst_node *left;    // left subtree with strings < str
  struct bst_node *right;   // right subtree with strings > str
} BST_Node, *BST_NodePtr;
```

Complete the implementation of `insert` on the next page so that `insert(str,r)` adds a copy of `str` to the BST with root `r` if `str` is not already present in the tree, and returns a pointer to the root of the modified tree. If `str` is already in the tree, `insert` should just return the pointer to the root of the original tree. For example, if we have a BST whose root node is `words`, the following code adds `"pqr"` to this BST if `"pqr"` is not already present.

```
words = insert("pqr", words);
```

Note that the tree (`words` in this example) may be empty (`NULL`) initially. When new strings are added, the code should preserve the binary search tree property that left subtrees have values less than their parent node, and right subtree values are greater. For full credit your answer should only traverse the necessary nodes in the tree.

You may assume that the original tree is properly formed, in particular, each node has a non-`NULL` `str` pointer and the string it points to is a properly '\0'-terminated array of characters (a C string) allocated on the heap. Also assume that all necessary library header files have already been `#include`d.

Hints: there are pages at the end of the exam with reference information that might be useful. If need to use `malloc` you may assume that it always succeeds and you do not need to check for errors.

Hint: Hint: Hint: recursion is your friend (yes, it really is!), as is your experience with BSTs in CSE143.

Write your answer on the next page and **remove this page from the exam. This page will not be scanned for grading.**

(continued on next page)

**Question 3 (cont.)** Write your implementation of function `insert` below.

```
// insert string s into the BST with root r if not present
// and return a pointer to the root of the updated tree
BST_NodePtr insert(char *s, BST_NodePtr r) {

  // If tree is empty, return a new node containing s
  if (r == NULL) {
    BST_NodePtr n = (BST_NodePtr)malloc(sizeof(BST_Node));
    n->str = (char*)malloc(strlen(s)+1);
    strcpy(n->str, s);
    n->left = NULL;
    n->right = NULL;
    return n;
  }

  // Tree is not empty.  If s is the same as the string in
  // the root node, return the root node without altering
  // the tree.  Otherwise insert the string in the left or
  // right subtree as appropriate and return the root of
  // the resulting tree.
  int cmp = strcmp(s, r->str);
  if (cmp < 0)
    r->left = insert(s, r->left);
  else if (cmp > 0)
    r->right = insert(s, r->right);
  return r;
}
```

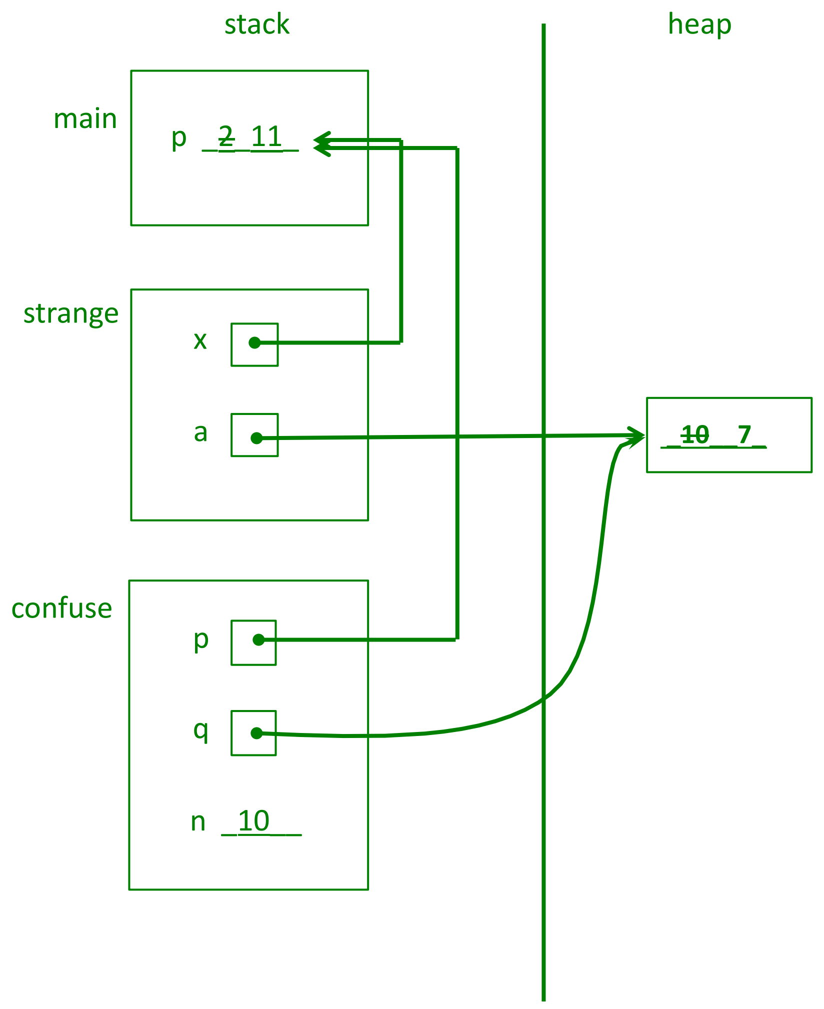
**Question 4.** (16 points) Memory madness. Consider the following program which, as traditional, does compile and execute successfully.

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

void confuse(int* p, int* q, int n) {
  *q = n-3;
  *p = n+1;
  ////HERE////
  printf("confuse: %d %d %d\n", *p, *q, n);
}

void strange(int* x) {
  int *a = (int*)malloc(sizeof(int));
  *a = 10;
  printf("before: %d %d\n", *x, *a);
  confuse(x, a, *a);
  printf("after: %d %d\n", *x, *a);
  free(a);
}

int main(int argc, char* argv[]) {
  int p;
  p = 2;
  strange(&p);
  printf("done: %d\n", p);
  return 0;
}
```

Answer questions about this program on the next page and **remove this page from the exam. This page will not be scanned for grading.**

(continued on next page)

**Question 4. (cont.)** (a) (10 points) Draw a boxes 'n arrows diagram showing state of memory when control reaches the comment containing `////HERE////` in the middle of function `confuse`. Your diagram should have three boxes showing the stack frames for functions `main`, `strange`, and `confuse`. The stack frames should show values of all local variables. Draw an arrow from each pointer to the location that it references. Data that is allocated on the heap should be drawn in a separate area, since it is not part of any function stack frame After drawing your diagram, be sure to answer part (b) at the bottom of the page.



(b) (6 points) What output does this program produce when it is executed?

```
before: 2 10
confuse: 11 7 10
after: 11 7
done: 11
```

**Question 5.** (16 points) Consider the following C++ program, which contains a class that represents a 4-digit PIN number for a bank ATM account.

```
#include <iostream>

class PinNumber{
public:
  PinNumber(const int first_digit, const int second_digit,
            const int third_digit, const int fourth_digit) {
    pin_ = new int[4];
    pin_[0] = first_digit;
    pin_[1] = second_digit;
    pin_[2] = third_digit;
    pin_[3] = fourth_digit;
  }
  ~PinNumber() { delete[] pin_; }
  // returns true if the other PinNumber is equal to this one
  bool equals(const PinNumber other) {
    return this->pin_[0] == other.pin_[0]
      && this->pin_[1] == other.pin_[1]
      && this->pin_[2] == other.pin_[2]
      && this->pin_[3] == other.pin_[3];
  }
private:
  int *pin_;
};

int main () {
  PinNumber* pin1 = new PinNumber(1, 2, 3, 4);
  PinNumber pin2(3, 4, 5, 6);
  bool equivalent = pin1->equals(pin2);
  if (equivalent) {
    std::cout << "Access given" << std::endl;
  } else {
    std::cout << "Access denied" << std::endl;
  }
  PinNumber pin3(11, 12, 13, 14);
  pin3 = pin2;
  if (pin2.equals(pin3)) {
    std::cout << "pins match" << std::endl;
  } else {
    std::cout << "something broken" << std::endl;
  }
  delete pin1;
}
```

Please write your answer on the next page and **remove this page from the exam. This page will not be scanned for grading.**

(continued on next page)

**Question 5.** (cont) (a) (6 points) This program compiles successfully, but when we try to run it there are a bunch of runtime errors. What's wrong? Give a brief, but precise technical description of the problem(s) and the reason(s) for the crash(es).

**The bug is that all copies of a single `PinNumber` object share a single array object because the default copy constructor and assignment operators for the class make a shallow copy of the object. That means that when any `PinNumber` destructor runs it deletes the `pin_` array that might be shared with another `PinNumber` object, leaving dangling pointers and causing eventual double-delete errors when the other `PinNumber` object(s) is(are) destroyed.**

(b) (10 points) What needs to be done to fix the problem(s)? If it involves adding or changing any code, write any new code below and precisely describe any other changes needed.

**Add a copy constructor that allocates a new array for each new object, and an assignment operator that copies the contents of the source array to the destination, instead of modifying the `pin_` pointer itself. Code:**

```
  // copy constructor
  PinNumber(const PinNumber &other) {
    pin_ = new int[4];
    pin_[0] = other.pin_[0];
    pin_[1] = other.pin_[1];
    pin_[2] = other.pin_[2];
    pin_[3] = other.pin_[3];
  }

  // assignment
  PinNumber &operator=(const PinNumber &other) {
    if (this == &other)
      return *this;
    // re-use pin_ array for new values
    pin_[0] = other.pin_[0];
    pin_[1] = other.pin_[1];
    pin_[2] = other.pin_[2];
    pin_[3] = other.pin_[3];
    return *this;
  }
```

**(It would also be possible in the assignment operator to delete the old array and allocate a new one, but there is no advantage to doing that in this case. Reusing the existing array saves the cost of the extra dynamic memory allocation operations.)**

**Question 6.** (12 points) Constructor madness. Consider the following (very unusual) C++ program which does compile and execute successfully. At the bottom of this page, write the output produced when it is executed.

Hints: remember that member variables are initialized in declaration order. Destruction order is the reverse of construction order. The body of a constructor runs after its initializer list. Last, notice that this program consists of a single object declaration and then termination, but that triggers a whole bunch of constructor and destructor calls.

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

class foo {
public:
  foo()                              { cout << "p"; }  // ctor
  foo(int i)                         { cout << "a"; }  // ctor(int)
  foo(int i, int j)                  { cout << "h"; }  // ctor(2int)
  foo(const foo &rhs)                { cout << "o"; }  // cctor
  foo &operator=(const foo &rhs)     { cout << "f"; return *this; }
  ~foo()                             { cout << "s"; }
};
class bar {
public:
  bar(): foo_(new foo())             { cout << "g"; }   // ctor
  bar(int i): foo_(new foo(i))       { cout << "p"; }   // ctor(int)
  bar(const bar &rhs)                { cout << "r"; }   // cctor
  bar &operator=(const bar &rhs)     { cout << "m"; return *this; }
  ~bar()                             { cout << "e"; delete foo_; }
private:
  foo *foo_;
  foo otherfoo_;
};
class baz {
public:
  baz(int a,int b,int c): foo_(b,c), bar_(a)
                                     { cout << "i"; }  // ctor(3int)
  baz(const baz &rhs)                { cout << "d"; }  // cctor
  baz &operator=(const baz &rhs)     { cout << "l"; return *this; }
  ~baz()                             { cout << "n"; }
private:
  foo foo_;
  bar bar_;
};
int main() {
  baz b(1,2,3);
  return 0;
}
```

Output: __happinesss___

**Question 7.** (12 points) A little programming including a function template. We're working on a C++ program and have discovered a function that needs to be implemented. The function is declared as follows in a header file:

```
// Given an array a with *len elements sorted in non-decreasing
// order, insert x into the array and increment *len so that the
// updated array a[0..*len-1] is sorted in non-decreasing order.
// Assume that the array is large enough to hold the new element.
void InsertSorted(int a[], uint32_t *len, int x);
```

So, for example, if an array `nums` contains `{-2,5,7}` and `size` contains 3 (the number of array elements), then calling `InsertSorted(nums, &size, 5);` should insert the new value in the proper place so that `nums` contains `{-2,5,5,7}`and `size` should be increased to 4.

For this problem, implement this function, with the following modification: change the function to be a template so that it will work with array element values of any type, provided that the array elements and the value `x` can all be compared to each other using the < (less than) operator. (i.e., `x` and the array elements will have the same type which could be anything that can be compared with < like `int`, `string`, `double`, etc.). Write your solution below. You can assume that any necessary library files have already been `#include`d. There is more room on the next page if you need it.

```
template <typename T>
void InsertSorted(T a[], uint32_t *len, T x) {
  // shift elements of a > x to right
  uint32_t k = *len;
  while (k > 0 && x < a[k-1]){
    a[k] = a[k-1];
    k--;
  }
  // store x in opened position and increment len
  a[k] = x;
  (*len)++;   // or *len=*len+1; parens required for ++
}
```

**Question 8.** (1 free point) (All reasonable answers receive the point. All answers are reasonable as long as there is an answer. ☺)

Draw a picture of something you did to pass the time while UW was closed because of the winter weather the last couple of weeks.