CSE 333 18au Midterm Exam Nov. 2, 2018
Sample Solution

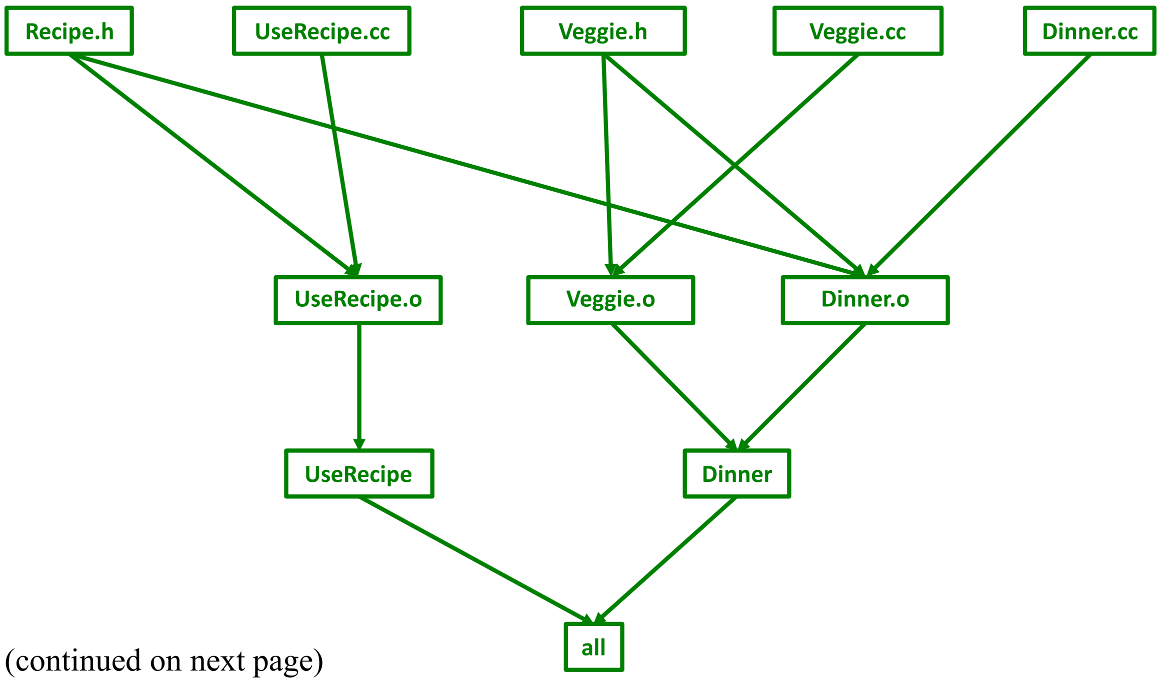
Question 1. (16 points) Build tools and `make`. We're building a C++ software back-end prototype for a new food web site. So far, we've got the following source files with the code for two main programs (`UseRecipe` and `Dinner`) and the other files that they use. (Header file guards omitted to save space, but assume they are there.)

```
==========
Veggie.h
==========
class Veggie { ... };

==========
Veggie.cc
==========
#include "Veggie.h"
// implementation of Veggie
//   class functions

==========
Recipe.h
==========
struct Recipe { ... }
```

```
==========
UseRecipe.cc
==========
#include "Recipe.h"

int main(...) { ... }

==========
Dinner.cc
==========
#include "Recipe.h"
#include "Veggie.h"

int main(...) { ... }
```

We want to construct a `Makefile` that has a default target named `all` that will build both of these programs. There should also be targets to build the individual `UseRecipe` and `Dinner` programs, and, when any source file is changed, only the necessary `.o` files and programs should be recompiled and/or relinked.

(a) (6 points) Draw the dependency diagram (dag) showing the build dependencies between the source files, the `.o` files they generate, and the programs built from them. Be sure to include the `all` target the builds everything.



(continued on next page)

Question 1. (cont.) (b) (8 points) Write an appropriate `Makefile` to build the program according to the dependency information given in your answer to part (a). The first couple of lines are written for you, giving the `all` target and a `CCFLAGS` variable that you can use in other rules in your answer. Remember, to include the contents of `CCFLAGS` in one of your rules, you can write `$(CCFLAGS)`. Your `Makefile` should build the program using the options specified in `CCFLAGS`.

In addition, there should be a `clean` target in the `Makefile` that will remove all of the executable programs, `.o` files created by the `Makefile`, and also remove any editor or other files whose filename ends with ~.

```
CCFLAGS = -Wall -g -std=c++11
all: UseRecipe Dinner

# Write the rest of the Makefile code below

UseRecipe: UseRecipe.o
     g++ $(CCFLAGS) -o UseRecipe UseRecipe.o

Dinner: Dinner.o Veggie.o
     g++ $(CCFLAGS) -o Dinner Dinner.o Veggie.o

UseRecipe.o: UseRecipe.cc Recipe.h
     g++ $(CCFLAGS) -c UseRecipe.cc

Dinner.o: Dinner.cc Recipe.h Veggie.h
     g++ $(CCFLAGS) -c Dinner.cc

Veggie.o: Veggie.cc Veggie.h
     g++ $(CCFLAGS) -c Veggie.cc

clean:
     rm UseRecipe Dinner *.o *~
```

(c) (2 points) Suppose we run `make` and build the program. Then, after that is done, someone makes a change to `Veggie.h`. List below the `Makefile` targets that are rebuild after this change if we run `make` again. The targets should be listed in the order that `make` will rebuild them.

1) `Dinner.o` and `Veggie.o` will be rebuilt first, but the order is not determined by the `Makefile`
2) `Dinner` will be relinked after that

Question 2. (12 points) The preprocessor, but using C++ this time! We have the following two files:

hdr.h:

```
#ifndef _HDR_H_
#define _HDR_H_

#define DBL(x) x * 2
typedef int number;
#define double float

#endif // _HDR_H_
```

ppro.cc:

```
#include <iostream>
#include "hdr.h"
#define PI 3.1416
using namespace std;
int main(int argc, char**argv){
  number a = PI;
  double b = 42.17;
  cout<< a << endl << b << endl;
  int n = 3;
  cout << DBL(n+1+1) << endl;
}
```

(a) (9 points) Show the result produced by the C/C++ preprocessor when it processes file `ppro.cc` (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 <iostream>` directive since that includes library declarations that we do not have access to. Write the rest of the preprocessor output below.

Hint: Although this is C++ code, it's the *same* preprocessor and it works exactly the same as it does for C programs.

```
typedef int number;
using namespace std;
int main(int argc, char**argv) {
  number a = 3.1416;
  float b = 42.17;
  cout<< a << endl << b << endl;
  int n = 3;
  cout << n+1+1 * 2 << endl;
}
```

(b) (3 points) What output does this program print when it is executed? (It does compile and execute without errors.)

```
3
42.17
6
```

Note that `number` is a `typedef` synonym for `int`, so `3.1416` is truncated to `3` when it is stored in `a`.

Question 3. (16 points) The nodes in a linked list of C strings can be defined as follows:

```
typedef struct strnode {
  char * str;            // this node's heap-allocated string
  struct strnode * next; // next node in the list or NULL
} Snode;
```

Complete the definition of function `Clone` below so that it returns (a pointer to) an exact duplicate of the list that is its argument, including duplicates of all the nodes and strings in the original list (i.e., a "deep copy"). You may assume that the original list 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). 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. You will need to use `malloc` to allocate nodes and strings for the copy. You may assume that `malloc` always succeeds and you do not need to check for errors.

```
// Return a clone of the linked list with first node lst
// (which might be NULL)
Snode * Clone(Snode *lst) {

  // Simple recursive solution

  // return NULL if at end of list
  if (lst == NULL) {
    return NULL;
  }

  // create a new node that is a clone of
  // the first node in the list
  Snode * dupnode = (Snode *)malloc(sizeof(Snode));
  dupnode->str = (char *)malloc(1+strlen(lst->str));
  strcpy(dupnode->str, lst->str);

  // next field of new node is clone of rest of the list
  dupnode->next = Clone(lst->next);

  // return new node, which is head of the cloned list
  return dupnode;

}
```

(more space on the next page for your answer if needed)

Question 3 (cont.) Additional space for your answer if needed.

Second solution using a loop. This one is arguably more space efficient since it does not have a nest of recursive calls. That means that the function itself needs constant space instead of space proportional to the number of nodes in the list for the function stack frames holding local parameters and variables.

```
Snode * Clone(Snode *lst) {
  if (lst == NULL) {
    return NULL;
  }
  // initialize head and tail of list created so far
  Snode *head = (Snode*)malloc(sizeof(Snode));
  Snode *tail = head;
   while (lst != NULL) {
    // copy the data from current lst node
    tail->str = (char*)malloc(1+strlen(lst->str));
    strcpy(tail->str, lst->str);
    // initialize tail->next and advance
    lst = lst->next;
    tail->next = (lst == NULL)? lst
                         : (Snode*)malloc(sizeof(Snode));
    tail = tail->next;
  }
  return head;
}
```

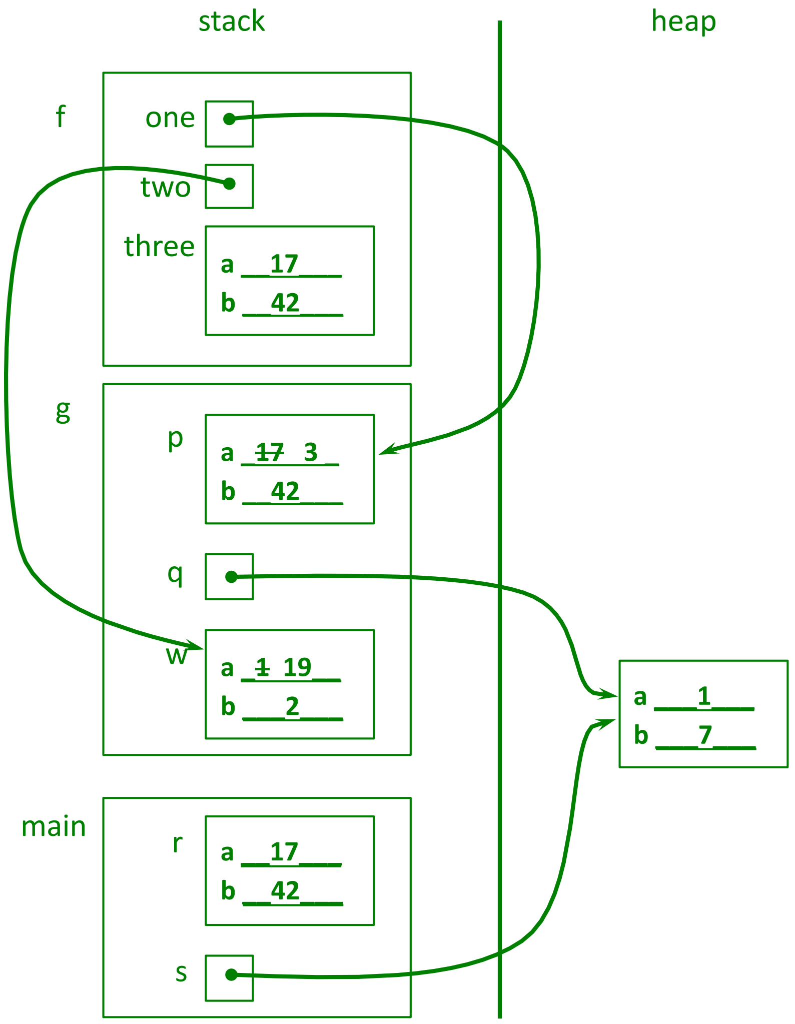
Question 4. (22 points) Memory madness. Consider the following program which, as traditional, does compile and execute successfully.

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

typedef struct pair {
  int a, b;
} Pair, *PairPtr;

void f(PairPtr one, PairPtr two, Pair three) {
  two->a = 2 + one->a;
  *one = three;
  one->a = two->b + 1;
  ////HERE//// (see question part a below)
  printf("*one = %d, %d; *two = %d, %d; three = %d, %d\n",
         one->a, one->b, two->a, two->b, three.a, three.b);
}

void g(Pair p, PairPtr q) {
  Pair w = {1, 2};
  f(&p, &w, p);
  printf("p = %d, %d; *q = %d, %d; w = %d, %d\n",
               p.a, p.b, q->a, q->b, w.a, w.b);
}

int main() {
  Pair r = {17, 42};
  PairPtr s = (PairPtr)malloc(sizeof(Pair));
  s->a = 1;
  s->b = 7;
  g(r,s);
  printf("r = %d, %d; s = %d, %d\n",
          r.a, r.b, s->a, s->b);
  free(s);
  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) (14 points) Draw a boxes 'n arrows diagram showing state of memory when control reaches the comment containing `////HERE////`in the middle of function `f`. Your diagram should have three boxes showing the stack frames for functions `main`, `f`, and `g`. The stack frames should show values of all local variables. Draw each pair struct as a box with two labeled fields `a` and `b`. 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) (8 points) What output does this program produce when it is executed?

***one = 3, 42; *two = 19, 2; three = 17, 42**
p = 3, 42; *q = 1, 7; w = 19, 2
r = 17, 42; s = 1, 7

Question 5. (16 points) Constructor madness. Consider the following C++ program which does compile and execute successfully. On the next page, write the output produced when it is executed.

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

static int idnum = 1;    // global var: next obj id number

class obj {
public:
  obj() {                    // default constructor
    id_ = idnum; idnum++;
    cout << "obj " << id_ << ": default constructor" << endl;
  }
  obj(int n) {               // int constructor
    id_ = idnum; idnum++;
    cout << "obj " << id_ << ": int constructor" << endl;
  }
  obj(const obj & other) {   // copy constructor
    id_ = idnum; idnum++;
    cout << "obj " << id_ << ": copy constructor from " <<
                                          other.id_ << endl;
  }
  obj& operator=(const obj & other) {  // assignment operator
    cout << "obj " << id_ << ": assignment operator from " <<
                                          other.id_ << endl;
    return *this;
  }
  ~obj() { // destructor
    cout << "obj " << id_ << ": destructor" << endl;
  }
private:
  int id_;    // this obj's id number
};

int main() {
  obj a;         // output is obj 1: default constructor
  obj b(a);
  obj c = 5;
  obj d = c;
  a = c;
  b = 5;
  cout << "done!" << endl;
}
```

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)

Queston 5 (cont.) On this page, write the output produced when the program from the previous page is executed. It does compile and execute successfully.

Note that when an object is constructed, the constructor stores a unique integer `id_` number, and operations on each object print out that object's `id_` number when they are executed. The first object's `id_` number is 1, and each new object has an `id_` number that is 1 greater than the previous object.

Also note that the constructors and assignment operations ignore their arguments. That, of course, would not happen in real code, but for this question it was done to save space since the values of the arguments are not needed to trace the program's execution.

The first output line is written for you. Write the rest of the program's output after that.

Output:

```
obj 1: default constructor
obj 2: copy constructor from 1
obj 3: int constructor
obj 4: copy constructor from 3
obj 1: assignment operator from 3
obj 5: int constructor
obj 2: assignment operator from 5
obj 5: destructor
done!
obj 4: destructor
obj 3: destructor
obj 2: destructor
obj 1: destructor
```

Note: for the assignment `b=5`, the compiler has to construct an `obj` temporary using the class `obj` `int` constructor, and then use that temporary object as the source value for the assignment to `b`. The compiler then generates code to automatically destroy the temporary. When we ran the program, the destructor code for the temporary executed right after the assignment, but it could have happened any time before the program terminated. Solutions that showed the destructor executing anywhere after the assignment received proper credit.

Question 6. (16 points) Trick or Treat! After a very successful Halloween night, you have a big bag of goodies. We'd like to figure out what was the most popular treat this year. But being computer geeks it would be too simple simply to count things: instead we need to write a C++ program to do it, especially now that we've learned about these fantastic C++ container libraries! ☺

Write a C++ program that reads from `cin` (standard input) a list of treats and writes on `cout` (standard output) the name of the most popular one. For instance, if the input is

```
Snickers  KitKat  Starburst  Skittles  KitKat  KitKat
apple  Skittles  banana
```

then the program should output the string `KitKat`, since that appears more often in the input than any other string. If there is a tie for the most popular treat, print any one of the winners (i.e., you can break ties however you like and should only print one answer).

Your answer should use C++ STL containers appropriately (`map` is likely to be especially useful). You do not need to `#include` any headers; assume that is done for you, and assume that there already is a `using namespace std;` line in the code to save some typing.

You should assume that if `s` is a C++ string, you can use `cin>>s` to read the treat names from standard input, and that each string read this way is a treat to be counted. Different strings represent different treats, so, for example, `chocolate`, `Chocolate`, and `CHOCOLATE` are all different since they don't match exactly.

You do not need to check for errors or unusual conditions on input other than detecting the end of input when it is reached. You may assume that there is at least one word in the input.

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)

Don't Write Here.

Go to the next page.

Thanks
|
|
v

Question 6 (cont.) Write the code for your answer below. A couple of lines are provided for you to get started

```
#include ... // assume all necessary libraries are included
using namespace std;
int main(int argc, char **argv) {
  // read treats from input and count how many of each
  map<string, int> m;
  string word;
  while (cin >> word) {
    ++m[word];
  }

  // find and print one treat with largest count
  pair<string, int> max = *m.begin();
  for (auto it = m.begin(); it != m.end(); ++it) {
    if (it->second > max.second) {
      max = *it;
    }
  }
  cout << max.first << endl;
  return 0;
}
```

There are obviously many other possible solutions. As with other questions, a correct solution received credit even if it doesn't match this one exactly.

Question 7. (2 free points) (All reasonable answers receive the points. All answers are reasonable as long as there is an answer. ☺)

(a) (1 point) What is your favorite `gdb` command? Either circle one of the commands listed below, or else write in your favorite if it isn't included in this list.

`break`	`enable`	`print`
`bt`	`exit`	`quit`
`catch`	`findbugs`	`return`
`checkstyle`	`finish`	`run`
`clear`	`fix`	`show`
`continue`	`help`	`step`
`debug`	`info`	`tui`
`disable`	`kill`	`until`
`disassem`	`list`	`up`
`display`	`make`	`watch`
`down`	`next`	`xyzzy`
`dwim`	`orkin®`	`zip`

Something else (what?) ________________________________

(b) (1 point) Explain (briefly) why this is your favorite `gdb` command: