CSE 333 24wi Midterm Exam 2/9/24 Sample Solution

Question 1. (18 points) Making things. We're working on a new app that recommends recipes based on information about things stored in the fridge. So far, we've got the following files with these #includes to reference declarations in various header files:

main.c	recipe.h	recipe.c

`#include "recipe.h"`
`...`
	`...`	`#include "recipe.h"`
		`#include "fridge.h”`
		`...`

fridge.c	fridge.h

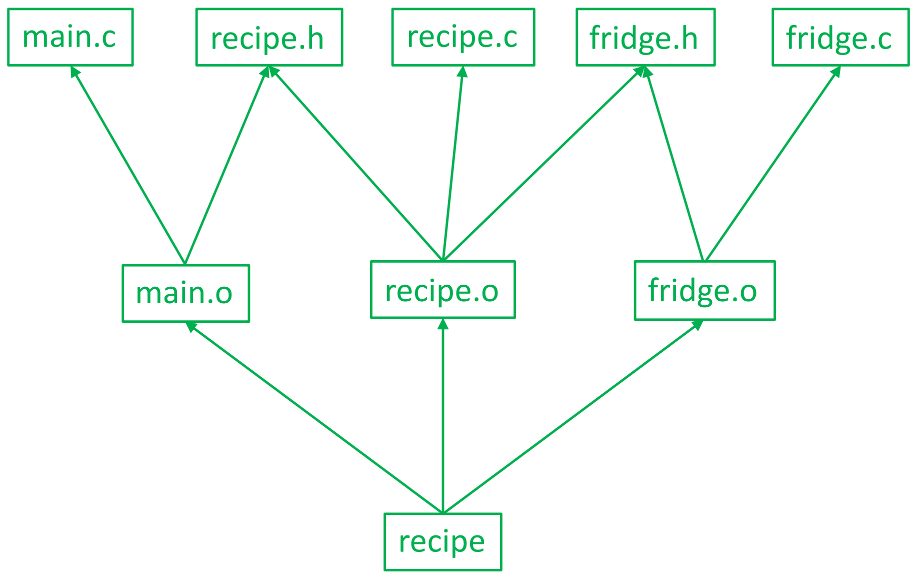
`#include "fridge.h"`
`...`
	`...`

We've been retyping the following `gcc` commands to build the program:

```
gcc -Wall -g --std=c17 -c fridge.c
gcc -Wall -g --std=c17 -c recipe.c
gcc -Wall -g --std=c17 -c main.c
gcc -Wall -g --std=c17 -o recipe main.o recipe.o fridge.o
```

Hint: recall that if we compile `foo.c` with the `-c` option and do not specify the output file name (no `-o` option), the output file created will be named `foo.o`, as in the `gcc` commands above.

(a) (6 points) Draw the dependency diagram that these commands and files represent (i.e., which files depend on which other files to build the final target program `recipe`?)



(continued on next page)

Question 1. (cont.) (b) (12 points) Write the contents of a `Makefile` for this program that has the following properties:

- The command `make` should build the program `recipe` from the source files using the `gcc` options `-Wall -g --std=c17`.
- The command `make` should only recompile and link files that need to be rebuilt after any changes to source files. Existing `.o` files and other files should not be re-compiled or relinked if that is not needed to bring the program up to date.
- You do not need to include anything else – no `clean` target or anything not mentioned above.

For reference, here are the tables of file information from the previous page:

main.c	recipe.h	recipe.c

#include "recipe.h"
...
	...	#include "recipe.h"
		#include "fridge.h"
		...

fridge.c	fridge.h
#include "fridge.h" ...	...

Write the `Makefile` code below.

```
recipe: main.o recipe.o fridge.o
	gcc -Wall -g --std=c17 -o recipe main.o recipe.o fridge.o
fridge.o: fridge.c fridge.h
	gcc -Wall -g --std=c17 -c fridge.c
recipe.o: recipe.c recipe.h fridge.h
	gcc -Wall -g --std=c17 -c recipe.c
main.o : main.c recipe.h
	gcc -Wall -g --std=c17 -c main.c
```

Note: the `recipe` target needs to be first to be the default target when one is not specified on the `make` command. The other targets can be listed in any order

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

hdr.h	prepro.c

`#ifndef HDR_H_`
`#define HDR_H_`
 	
`#define dbl double`
`typedef long int longint;`
`#define IT 42`
`#define THAT 12 + IT`
`#define OTHER IT + THING`
 	
`#endif`
	`#include "hdr.h"`
	
	`#define THING 333`
	
	`int fun(dbl x, longint n) {`
	`  int k = x + THING;`
	`  double y = THAT + IT;`
	`  n += 12;`
	`  return n + OTHER;`
	`}`

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?)

```
typedef long int longint;

int fun(double x, longint n) {
  int k = x + 333;
  double y = 12 + 42 + 42;
  n += 12;
  return n + 42 + 333;
}
```

Question 3. (18 points) HW1 linked lists and hash tables. We would like to add a new public function to the `HashTable`s we created in hw1 to report the total number of buckets in a hash table, and the number of those buckets that contain no entries. The specification of this function is:

```
// Return the total number of buckets in HashTable ht in
// *nbuckets, and return the number of buckets that have no
// elements (are empty) in *nempty.
void HashTable_Bucket_Info(HashTable* ht,
                                  int *nbuckets, int *nempty);
```

Give an implementation of this function as it would appear in `HashTable.c`. Your code should only use the public interface to the `LinkedList` module since the `HashTable` implementation is a client of that module. Copies of the header files `LinkedList.h`, `HashTable.h`, and `HashTable_priv.h` are included at the end of the exam, and you should remove them from the exam to use for this question. They will not be scanned for grading. If needed, your code may allocate new data structures while it is executing, but should not allocate more data than needed and should not cause any memory leaks. You may assume that any linked list or hash table functions that you call will succeed, and you do not need to check for errors when you do that.

Write your solution below. (Hints: you probably won't need all of this space. The sample solution is about 8-10 lines long, but you don't need to match that.)

```
void HashTable_Bucket_Info(HashTable* ht,
                                  int *nbuckets, int *nempty) {
  *nempty = 0;
  *nbuckets = ht->num_buckets;
  for (int i = 0 ; i < ht->num_buckets; i++) {
    LinkedList *ll = ht->buckets[i];
    if (LinkedList_NumElements(ll) == 0) {
      *nempty += 1;
    }
  }
}
```

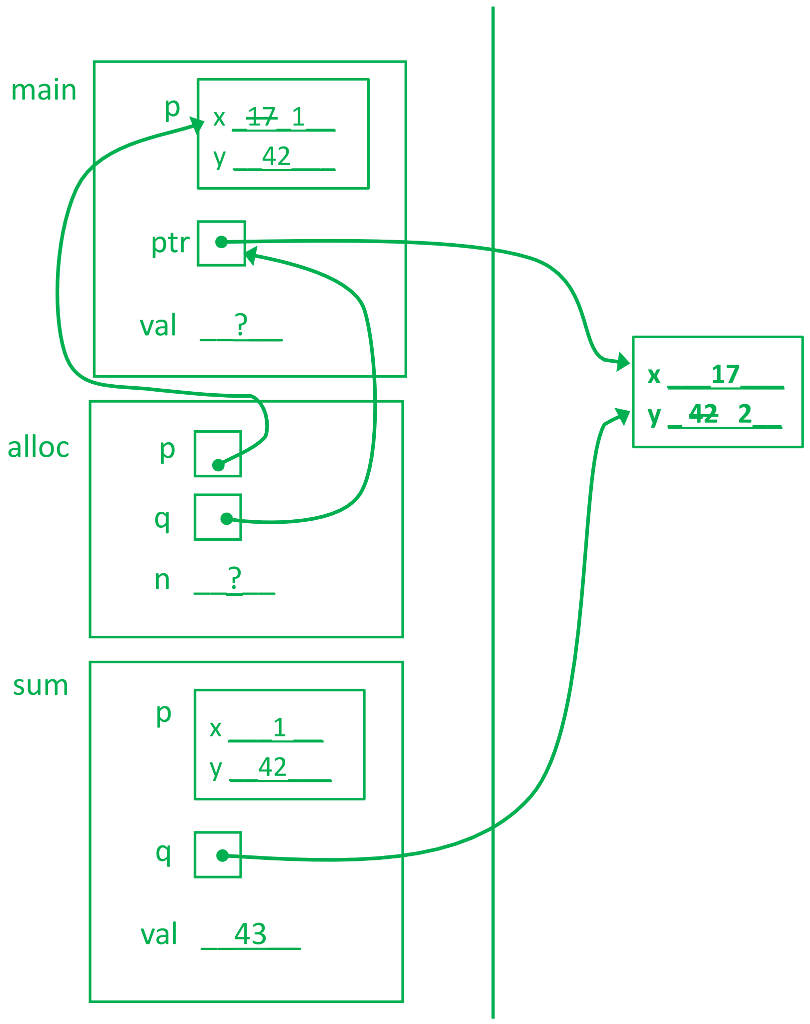
Question 4. (20 points) Pointer trickery. Here is a small C program that compiles and executes without errors, except that it leaks memory, which is not relevant for this question. Answer questions about it on the next page.

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

typedef struct pr {
  int x, y;
} Pair;

int sum(Pair p, Pair *q) {
  int val = p.x + p.y;
  q->y = 2;
  ///// draw memory diagram when control reaches here /////
  return val;
}

int alloc(Pair *p, Pair** q) {
  *q = (Pair *) malloc(sizeof(Pair));
  **q = *p;
  p->x = 1;
  int n = sum(*p, *q);
  return n;
}

int main(int argc, char **argv) {
  Pair p = {17, 42};
  Pair *ptr;
  int val;
  val = alloc(&p, &ptr);
  printf("%d %d\n", p.x, p.y);
  printf("%d %d\n", ptr->x, ptr->y);
  printf("%d\n", val);
  return EXIT_SUCCESS;
}
```

(continued on next page)

Question 4. (cont.) (a) (14 points) In the space below, draw a careful diagram showing the state of memory when execution reaches the point in function `sum` labeled "draw memory diagram when control reaches here". Be sure to carefully distinguish between local variables on the stack to the left and values allocated on the heap. Also be sure you clearly show the difference between pointers to structs and struct values that have x and y components. Local variables for each function should be drawn in a separate box for each function labeled with the function name. When you're done, be sure to answer the "what does it print" part of the question at the bottom.

Stack (local variables) Heap (dynamic storage – malloc)



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

```
1   42
17   2
43
```

Question 5. (20 points) C++ constructors and things. Here's a small C++ program that compiles and executes without errors (`#includes` omitted to save space).

```
class Toy {
 public:
  Toy() : kind_("ball")      { cout << "ball" << endl;}
  Toy(string t) : kind_(t)   { cout << kind_ << endl;}
  Toy(const Toy &other) : kind_(other.kind_)
                             { cout << "another " << kind_ << endl;}
  Toy &operator=(const Toy &other) {
    if (this != &other) {
      kind_ = other.kind_;
    }
    cout << "change to " << kind_ << endl;;
    return *this;
  }
  void dbl() { kind_ = kind_ + " " + kind_;
               cout << "double " << kind_ << endl; }
  ~Toy()  { cout << "gone " << kind_ << endl; }
 private:
  string kind_;
};

int main() {
  Toy mine;
  Toy yours("car");
  Toy* scooter = new Toy("scooter");
  mine = *scooter;
  delete scooter;
  Toy thing(yours);
  yours.dbl();
  return EXIT_SUCCESS;
}
```

What output does this program produce when it is executed? (It does compile and run without errors.)

```
ball
car
scooter
change to scooter
gone scooter
another car
double car car
gone car
gone car car
gone scooter
```

Question 6. (6 points) Recall the small C++ string class from lecture. `Str.h` defines the class as follows:

```
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
  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'
};
```

The assignment operator for this class is a bit complicated, reallocating the heap array that belongs to the `Str` each time we need to update a `Str` object:

```
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;
}
```

One of our colleagues has suggested that the code would be much more efficient if we simply updated the `st_` instance variable to point to the data array belonging to the other string, which contains the characters we want to assign. In other words, replace the entire body of the `operator=` function with the following two lines:

```
  st_ = s.st_
  return *this;
```

The question is, would this be a good idea and work properly? Answer yes or no and give a precise technical reason for your answer. (Hint: think about the other operations, constructors, and destructor in the class and how they need to use the `st_` variable.)

No.

The destructor for Str has to execute `delete [] st_` to avoid a memory leak. Once we've used the modified assignment operator, the `st_` variables in more than one `Str` object can point to the same heap array and the destructors in the `Str` objects will all delete the array, resulting in a double-delete error. There's no real way to avoid the problem since there's no way to tell if the array referenced by a st_ variable is also referenced by another pointer in another object or elsewhere.

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 question were you expecting to appear on this exam that wasn't included?

Implement a C compiler. Be sure to include all of the libraries that are part of the standard C language.

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

Yes

No

Heck No!!

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

[✓ Yes, yes, it *must* be included!!!]

No opinion / don't care

None of the above. My answer is ___________________________________.