University of Washington – Computer Science & Engineering

CSE 333 A Autumn 2023 Midterm: Version A

Last Name:	Sample
First Name:	Solutions
Student ID Number:	
UWNetID:	
All work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CSE333 who haven't taken it yet. Violation of these terms could result in a failing grade. **(please sign)**	

Do not turn the page until 11:30.

Instructions

- This exam contains 8 pages, including this cover page. Put your final answers in the boxes and blanks provided. You may make use of the 'overflow box' on the last page for additional answer space.
- The exam is closed book (no laptops, tablets, wearable devices, or calculators). You are allowed one 3"x5" index card (double-sided) of *handwritten* notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- You have 50 minutes to complete this exam.

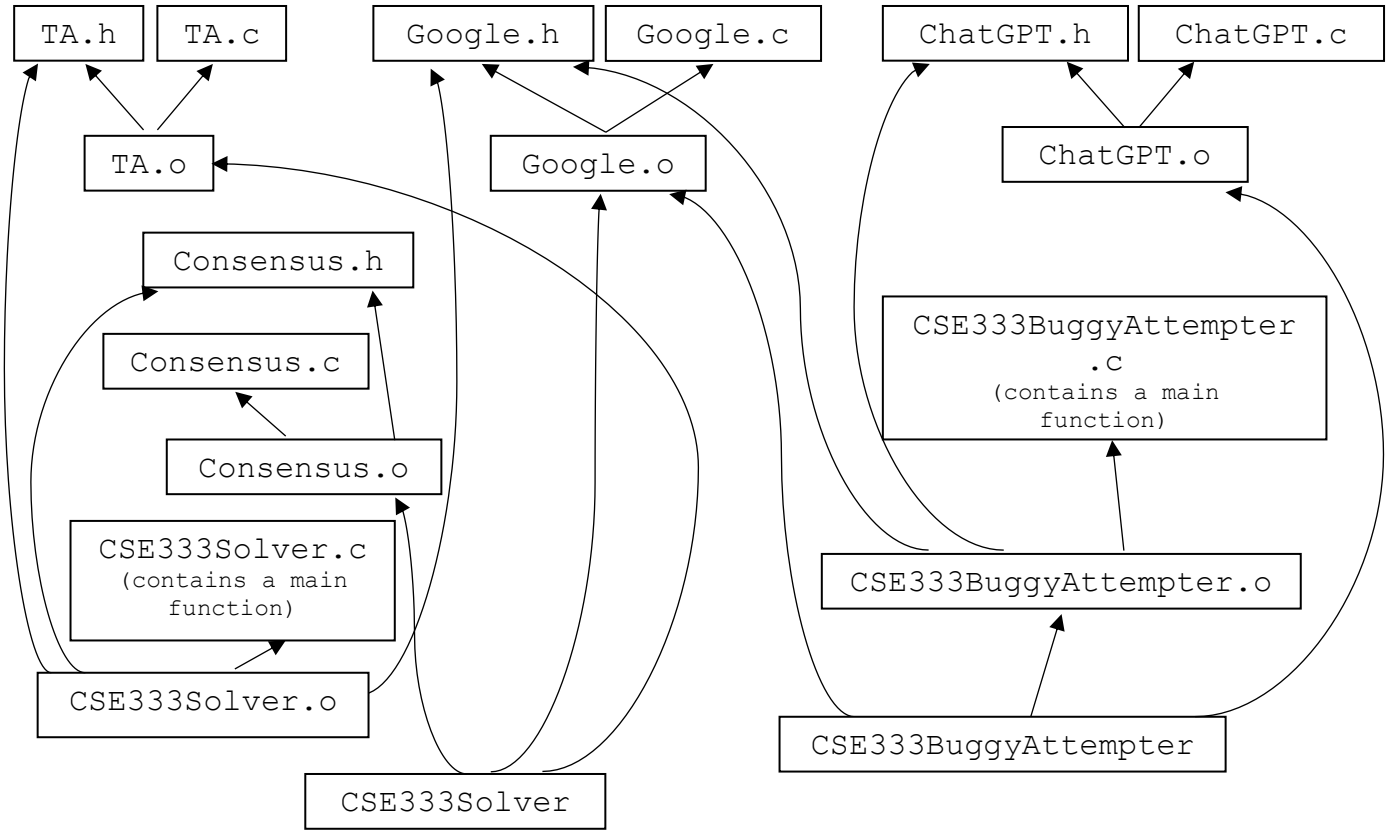
Advice

- Read questions carefully before starting. Skip questions that are taking a long time.
- Read *all* questions first and start where you feel the most confident.
- Relax. You are here to learn.

Question	1	2	3	4	5	6	Total
Possible Points	16	16	8	10	~~24~~ 22	1	~~75~~ 73

Question 1: Makefile of Many.

Consider the dependency graph, below, derived from a `Makefile` intended to build two executables that attempt to answer homework in CSE333 using different strategies. Note that in this representation arcs are drawn from targets to dependencies. Source files that contain a main function are those that contain the text "(contains a main function)" in the graph below.



(A) [3 pts] If `Google.h` is modified, which targets need to be rebuilt?

Google.o, CSE333BuggyAttempter.o, CSE333Solver.o
CSE333BuggyAttempter, CSE333Solver

(B) [3 pts] If `Google.c` is modified, which targets need to be rebuilt?

Google.o, CSE333BuggyAttempter, CSE333Solver

(C) [4 pts]. The `Makefile` snippet that generates the executable `CSE333BuggyAttempter` is below.

```
CSE333BuggyAttempter: CSE333BuggyAttempter.o Google.o ChatGPT.o
        $(CC) $(CFLAGS) -o $@ $^
```

Notes: Recall that `$@` is special syntax that will expand to the name of the current target and `$^` is special syntax that will expand to all listed sources of the current target.

Write a similar snippet for the executable target `CSE333Solver` based on the dependency graph above. You may choose to use the special syntax demonstrated in the previous snippet, or list out names of targets and sources explicitly.

```
CSE333Solver: Consensus.o Google.o TA.o
        $(CC) $(CFLAGS) -o $@ $^
OR:
CSE333Solver: Consensus.o Google.o TA.o
        $(CC) $(CFLAGS) -o CSE333Solver Consensus.o Google.o TA.o
```

(D) [3 pts] All object file targets correctly build in your `Makefile`. When running "`make CSE333BuggyAttempter`" the executable `CSE333BuggyAttempter` successfully builds. When running "`make CSE333Solver`" the executable `CSE333Solver` does not. Why?

```
The CSE333Solver target does not compile and/or link against any
code with a `main` function (executable entry point).
```

(E) [3 pts] The unsuccessful build is only due to an error in your Makefile (due to a missing relationship in dependency graph above). Rewrite any target(s) of your `Makefile` below that will result in the successful build of the `CSE333Solver` executable when running "`make CSE333Solver`". If no changes are necessary then check the box below.

☐ No changes necessary in the Makefile

```
CSE333Solver: Consensus.o Google.o TA.o CSE333Solver.o
        $(CC) $(CFLAGS) -o $@ $^
OR:
CSE333Solver: Consensus.o Google.o TA.o CSE333Solver.o
        $(CC) $(CFLAGS) -o CSE333Solver Consensus.o Google.o TA.o
```

Question 2: Perilous Pointer Puzzler.

Notes: This C++ code compiles and runs without error. A `char` is `1` byte.

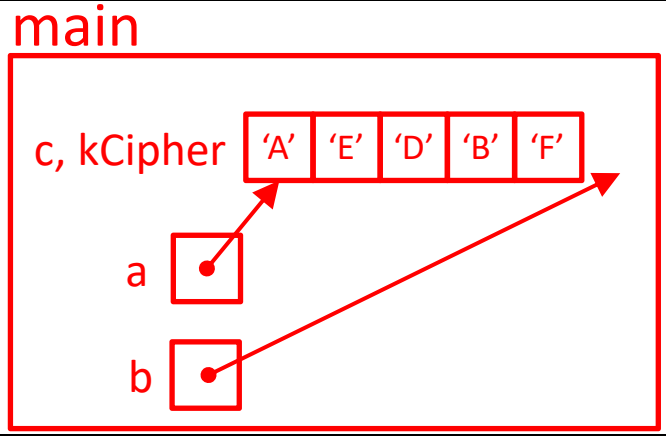
pointer_puzzle.cc

```
#include <iostream>

char F1(const char *c) { return *c; }
char F2(const char t[]) { return t[1]; }

int main(int argc, char *argv[]) {
    const char kCipher[] = {'A', 'E', 'D', 'B', 'F'};
    const char *a = &kCipher[0];
    const char *b = a + 3;
    const char &c = *a;
    char (*fp)(const char[]) = F2;
    a += 2;
    std::cout << F1(&kCipher[2]) << *(b - 2) << c << F2(kCipher + 3);
    a += 1;
    b += 2;
    std::cout << F1(*(&a)) << (a - kCipher) << c << fp(a);
    a = kCipher;
    // *** HERE ***
    return 0;
}
```

(A) [8 pts] Draw a memory diagram (*i.e.,* a boxes 'n arrows diagram) showing the state of memory when control reaches the comment containing "`*** HERE ***`". Your diagram should have boxes showing the stack frames for all active functions. The stack frames should show values of all local variables and denote any aliases. Draw an arrow from each pointer to the location that it references. You do not need to draw any function pointers in your diagram. If there is any data that is allocated on the heap, it should be drawn in a separate area, since it is not part of any function stack frame.



(B) [8 pts] What output is written to the console when the above program completes its execution?

DEAFB3AF

Question 3: POSIX.

[8 pts] Of the following, which are POSIX system calls and which are not?

	Syscall	Not Syscall
`size_t fread(const void *ptr, size_t size, size_t nmemb, FILE *stream);`		X
`struct dirent* readdir(DIR *dirp);`	X	
`void* malloc(size_t size);`		X
`int close(int fildes);`	X	

Question 4: PREPROCESSOR.

Notes: The `'%d'` format specifier for `printf` is used to format integers.

preprocess.c

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

#define PLUSONE(x) x + 1
#ifdef DDEBUG
#define OFFSET 4
#else
#define OFFSET 0
#endif

int main(int argc, char *argv[]) {
    int x = PLUSONE(2);
    int y = PLUSONE(40 + OFFSET);
    printf("%d, %d", x, y);
    return EXIT_SUCCESS;
}
```

(A) [5 pts] When `preprocess.c` is compiled with the command

```
gcc -Wall -std=c11 -DDEBUG -o preprocess preprocess.c
```

what does the executable `preprocess` output when run?

3, 41

(B) [5 pts] When `preprocess.c` is compiled with the command

```
gcc -Wall -std=c11 -o preprocess preprocess.c
```

what does the executable `preprocess` output when run?

3, 41

Question 5: Constructor Chaos.

Notes: This C++ code compiles and runs without error.

interval.h

```
#ifndef INTERVAL_H_
#define INTERVAL_H_

class Interval {
public:
  Interval(int a, int b);
  Interval(const Interval& o);
  ~Interval();
  Interval& operator=(const Interval& o);
  bool Contains(const Interval& o) const;
  int Diameter() const;
  void set_a(int a);
private:
 int a_;
 int b_;
};

#endif // INTERVAL_H_
```

ducks.cc

```
#include "interval.h"
#include <iostream>

int main(int argc, char *argv[]) {
    Interval a(1, 4);
    Interval b = a;
    b.set_a(4);
    Interval &c = a;
    std::cout << b.Diameter();
    b = c;
    if (!b.Contains(a)) {
        std::cout << "N";
    }
    // *** HERE ***
    return 0;
}
```

interval.cc

```
#include "interval.h"
#include <iostream>

Interval::Interval(int a, int b) : a_(a), b_(b) {
    std::cout << "B";
}

Interval::Interval(const Interval& o) : a_(o.a_), b_(o.b_) {
    std::cout << "ow";
}

Interval::~Interval() { std::cout << "!"; }

Interval& Interval::operator=(const Interval& o) {
    // Dragons be here!!
    std::cout << "W";
    return *this;
}

bool Interval::Contains(const Interval& o) const {
    return (a_ <= o.a_ && b_ >= o.b_);
}

int Interval::Diameter() const {
    if (a_ <= b_) return b_ - a_;
    return a_ - b_;
}

void Interval::set_a(int a) {
    std::cout << "D";
    a_ = a;
}
```

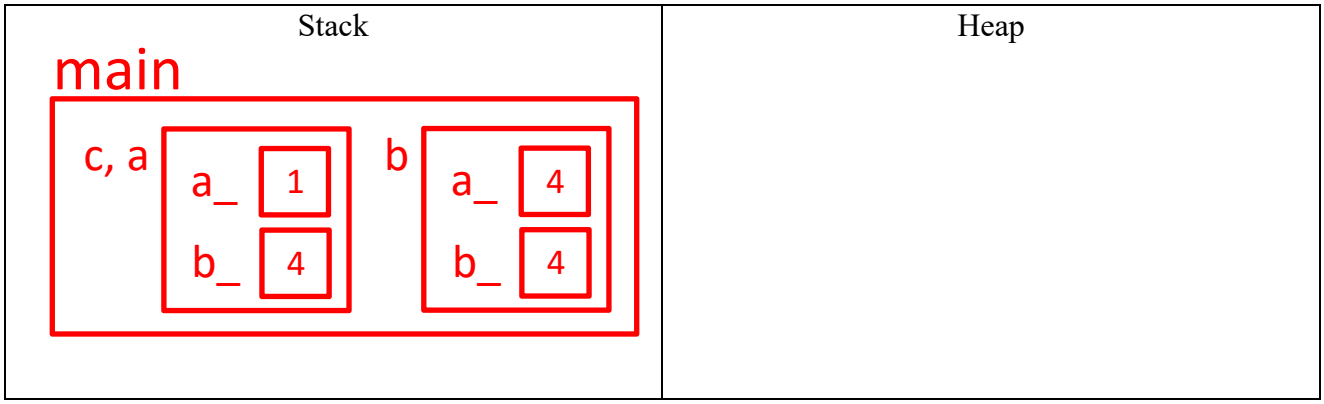
(A) [8 pts] This code successfully compiles on `attu` with the command `g++ -o ducks ducks.cc interval.cc`. When the executable `ducks` is run and control reaches the line marked "`*** HERE ***`" how many times have the following methods been invoked on objects of type `Interval`?

`Default Constructor`	`0`
`Copy Constructor`	`1`
`Destructor`	`0`
`Assignment Operator`	`1`

(B) [8 pts] What is printed to the console by the executable `ducks`?

```
BowD0WN!!
```

(C) [6 pts] Draw a memory diagram (*i.e.,* a boxes 'n arrows diagram) showing the state of memory when control reaches the line marked "`*** HERE ***`".



[0 pts]

(D) ~~[2 pts]~~ The assignment operator of `Interval` has a warning in a comment. Use the space below to either rewrite the definition of `Interval::operator=` or justify that it does not need to change.

+1 bonus: recognizes data members are not updated (via code rewrite or explanation).

+1 bonus: either of the following (via code rewrite or explanation)

- recognizes best practice is to check for self-assignment in `operator=`
- justifies this check is unnecessary in this case either because data members are not updated or because data members do not manage dynamic resources (e.g., point to heap allocated memory, or file descriptor, etc, or other dynamic resource this class is responsible for).

Question 6:

[1 pt; All non-empty answers receive this point] Select one member of the course staff. Describe or draw an emoji representing that person.

Any non-blank answer received 1pt.

Extra Work ("Overflow Box"):

You may put additional answers here so long as you indicate which question the work belongs to and indicate in the original answer box that you put an answer in the 'overflow box'.