CS 430 Types Lab

THIS IS THE TEMPLATE -- PLEASE MAKE A COPY AND REMOVE THIS LINE

NAME(S)

Part 1: Pointer Types

Consider the following C program:

```
int main() {
    int a = 123;
    int *b = &a;
    int **c = &b;
    // location A
    *b = 321;
    printf("%d %d %d\n", a, *b, **c);
    return 0;
}
```

1. On paper, draw a picture of stack memory at location A. Label memory locations with their C variable names, and draw pointers as arrows pointing to their targets. **(You do not need to turn in this diagram.)**

2. What is the name of the `*` operator in C (e.g., on the two lines after location A), and what does it do?

   ANSWER

3. Think about how your diagram would change after the line following location A. What is the program's output? (If the output cannot be determined, write "N/A".)

   ANSWER

4. Load PythonTutor.com/visualize.html and enter the above code (or use this shortcut: https://goo.gl/vm8BVg). Run the code (make sure you're in C mode) and confirm your answers to #1 and #3 above.

Consider the following C program:

```
int main() {
    char msg[] = "Hello!";
    msg[1] = 'o';
    char *p = msg;
    // location B
    *p = 'P';
    p++;
    *(p+4) = '?';
    printf("%s\n", msg);
    return 0;
}
```

5. Examine memory at location B using PythonTutor, and observe how the contents of memory change after each of the three lines following location B. (shortcut: https://goo.gl/W5HsnH)

6. What is the program output? (if the output cannot be determined, write "N/A")

   ANSWER

Part 2: Heap Pointers

Consider the following C program:

```
int main() {
    int *a = (int*)malloc(5*sizeof(int));
    int b = 2;
    int *p = &a[1];
    // location C
    a[0] = 777;
    a[b] = 45;
    *p = 9;
    p[2] = 12;
    p += b;
    *(p+1) = *a;
    // location D
    free(a);
    printf("%d\n", *p);
    return 0;
}
```

7. Examine memory at location C using PythonTutor, and observe how the contents of memory change between location C and location D. (shortcut: https://goo.gl/ZYqojG)

8. What is the program output? (if the output is undefined, write "N/A")

   ANSWER

9. Does the program leak memory? Are there any dangling pointers?

   ANSWER

10. Describe briefly (in 1-2 sentences) how the locks & keys approach described in CPL section 6.11.8.2 would help deal with a problem identified in #9.

    ANSWER

11. Describe briefly (in 1-2 sentences) how mark-sweep garbage collection would prevent a problem identified in #9.

    ANSWER

12. Describe briefly (in 1-2 sentences) how reference counters would prevent a problem identified in #9. How many references are there to each 4-byte memory cell allocated on the heap at location C?

    ANSWER

13. What is a reference type and how is it different from a pointer type? Does C have reference types, pointer types, or both? What about C++, Java, and Ruby?

    ANSWER

Part 3: Struct and Union Types

Consider the following C program:

```
struct {
  int a;
  char b;
  double c;
} v;

int main() {
    v.a = 7;
    v.b = 'x';
    v.c = 1.23;
    // location E
    printf("%d %c %f\n", v.a, v.b, v.c);
    return 0;
}
```

14. Examine memory at location E using PythonTutor. (shortcut: https://goo.gl/qbUpVy)

15. What is the minimum number of bytes necessary to store the struct variable ("v") in the above program? (assume that characters require 1 byte, integers require 4 bytes, and double-precision floating point numbers require 8 bytes)

    ANSWER

16. Suppose that on a particular system there are X different possible integers, Y different possible characters, and Z different possible double-precision numbers. As a function of X, Y, and Z, what is the total number of different possible values for the struct variable ("v")?

    ANSWER

17. Suppose that we changed "struct" to "union" in the first line. Consider how this would change what is stored in memory and what effect it would have on the output. What is the minimum number of bytes required to store the union variable?

    ANSWER

18. As a function of X, Y, and Z (as defined in #16), what is the total number of different possible values for the union variable ("v")?

    ANSWER

NOTE: PythonTutor doesn't appear to have support for union types, so if you wish to execute the modified program you will have to compile and run the code yourself.

BONUS: Trace the code and draw memory for the following programs:

- https://goo.gl/ohJ1DB
- https://goo.gl/RTimVH
- https://goo.gl/JNDZdD
- https://goo.gl/UnqT8V
- https://goo.gl/eYMmHQ
- https://goo.gl/DTdy2j

Part 4: Type Equivalence

Suppose the following declarations have been made in a C-like language. For each context, **make bold** all of the assignments that are valid, and ~~strike through~~ all that are not valid (these formatting options are in the Format -> Text menu).

```
typedef float inches;
typedef float feet;
typedef struct { inches x; feet y; } box;
typedef struct { inches x; feet y; } bin;

inches a, b;
feet c;
float d;
struct { inches x; feet y; } m, n;
box o;
bin p;
```

19. Assume assignments require name equivalence unless at least one type is anonymous, in which case they only require structure equivalence.

a = b	a = c	a = d	a = m
m = n	m = o	p = o	m.x = d
m.x = a	o.x = a	m.x = m.y	m.x = o.x

20. Assume assignments only require structure equivalence, regardless of type aliases.

a = b	a = c	a = d	a = m
m = n	m = o	p = o	m.x = d
m.x = a	o.x = a	m.x = m.y	m.x = o.x

21. Assume assignments require name equivalence for primitive types and their aliases but permit structure equivalence for non-primitive types.

```
a = b           a = c           a = d           a = m

m = n           m = o           p = o           m.x = d

m.x = a         o.x = a         m.x = m.y       m.x = o.x
```