Name: ID#: X500: @umn.edu **A**

CS 2021: Practice Final Exam

Spring 2019
University of Minnesota

Exam period: 20 minutes
Points available: 40

Problem 1 (10 pts): There are two terminal sessions below both with errors in getting a program running. Describe in each case whether the problem is a (1) Compile error (2) Static Linking Error or (3) Dynamic Linker/Loader Error, and **how to fix the problem**.

```
# SESSION A
> gcc tree_main.c -I mylibraries/ -lbstree
/usr/bin/ld: cannot find -lbstree
collect2: error: ld returned 1 exit status
```

Type of Error and Fix for it:

```
# SESSION B
> gcc math_main.c -lm
math_main.c: In function ‘main’:
math_main.c:3:14: error:
  implicit declaration of function ‘pow’
     double x = pow(3.14, 4.6);
```

Type of Error and Fix for it:

Background: To the right is the output of `pmap` showing page table virtual memory mapping information for a running program called `memory_parts`. Answer the following questions about this output.

Problem 2 (5 pts): The mapped memory references something called `libc-2.26.so`. Describe this entity and what kind of information you would expect to find at the mapped locations.

Problem 3 (5 pts): Why does `pmap` only show a limited number of virtual addresses? What would happen if the program attempted to access an address not listed in the output? Example: address `0x00` is not in the listing.

```
pmap 7986
7986:   ./memory_parts
00005579a4abd000      4K r-x-- memory_parts
00005579a4cbd000      4K r---- memory_parts
00005579a4cbe000      4K rw--- memory_parts
00005579a4cbf000      4K rw---   [ anon ]
00005579a53aa000    132K rw---   [ heap ]
00007f441f2e1000   1720K r-x-- libc-2.26.so
00007f441f48f000   2044K ----- libc-2.26.so
00007f441f68e000     16K r---- libc-2.26.so
00007f441f692000      8K rw--- libc-2.26.so
00007f441f694000     16K rw---   [ anon ]
00007f441f698000    148K r-x-- ld-2.26.so
00007f441f88f000      8K rw---   [ anon ]
00007f441f8bb000      4K r---- gettysburg.txt
00007f441f8bc000      4K r---- ld-2.26.so
00007f441f8bd000      4K rw--- ld-2.26.so
00007f441f8be000      4K rw---   [ anon ]
00007fff96ae1000    132K rw---   [ stack ]
00007fff96b48000     12K r----   [ anon ]
00007fff96b4b000      8K r-x--   [ anon ]
 total             4276K
```

Problem 4 (20 pts): Below are two functions that augment El Malloc with the block shrinking; this allows a user to specify that the originally requested size for a memory area can be adjusted down potentially creating open space. Fill in the definitions for these functions.

```
el_blockhead_t *el_shrink_block(el_blockhead_t *head, size_t newsize){
// Shrinks the size of the given block potentially creating a new block.  Computes remaining space
// as the difference between the current size and parameter newsize. If this is smaller than
// EL_BLOCK_OVERHEAD, does nothing further and returns NULL. Otherwise, reduces the size of the
// given block by adjusting its header and footer and establishes a new block above it with
// remaining space beyond the block overhead. Returns a pointer to the newly introduced blocks. Does
// not modify any links in lists.
















}
int el_shrink(void *ptr, size_t newsize){
// Shrink the area associated with the given ptr if possible.  Checks to ensure that the block
// associated with the given user ptr is EL_USED and exits if not.  Uses el_shrink_block() to
// adjust the block size and create a block for the remaining space.  If not possible to shrink,
// returns 0.  Otherwise moves the current block to the front of the Used List and places the newly
// created block to the front of the Available List after setting its state to EL_AVAILABLE. Returns
// 1 on successfully shrinking.
















}
```