# 1 Administrivia

Homeworks should be done individually and turned in via Moodle. Homeworks will be graded with a percentage of completeness; the final homework grade (20% of total) will be calculated as an average of all homeworks, however many there may be.

This homework is worth 200 points.

# 2 Assignment

## 2.1 UNIX Warmup

1. (20 pts) On `os.cs.siue.edu`, create a subdirectory in your home directory called `hw0`, inside `hw0` create a subdirectory for each of the problems, using the numbering below to name the subdirectories. You should end up with directory structure like hw0/1, hw0/2, etc. Place each solution in its respective subdirectory.

2. (30 pts) All attempted problems MUST be done in C, MUST compile and run on os.cs.siue.edu, and MUST use pointers. Do not modify the prototypes.

3. (30 pts) You must include a makefile, which builds all solutions by default (e.g. simply typing `make`), but also allows for each solution to be built separately by designating the problem number (e.g. `make 1`).
   Hint: `https://www.gnu.org/software/make/manual/html_node/Phony-Targets.html`

4. (10 pts) When the problem asks for a function, include all code required for correct compilation and one or more test cases illustrating correct execution. If the test cases are not straightforward, include a plain text README file in the solution's subdirectory that explains the test case, usage, compilation, etc.

5. (10 pts) Solutions are correctly submitted as a single compressed tarball.

## 2.2 10 Easy Pieces

1. (10pts) Write a function `set(int* a, int val)` that sets the value of an integer a declared in the calling function, to the value `val`. Print the value of the integer from the calling function both prior to and after calling the `set` function.

2. (10pts) Write the function `swap_ints(int*, int*)` that swaps the values of two integers. From the calling function, print the values of two pointers before and after the swap.

3. (10pts) Write the function `swap_ptrs(int**, int**)` that swaps two pointers. From the calling function, print the values of the two integers before and after the swap.

4. (10pts) Implement a singly-linked list of integers. Write the function `find_Nth_element(struct listnode* head, int N)` which returns the $N^{th}$ element of the list. Print the contents of your list and the value of the $N^{th}$ element on separate lines.

5. (10pts) Implement a singly-linked list of integers. Write the function `remove_Nth_element(struct listnode* head, int N)` which removes the $N^{th}$ element of the list. Print the contents of the list before and after removal of the $N^{th}$ element.

6. (10pts) Implement a singly-linked list of integers. Write the function `find_Nth_to_last_element(struct listnode* head, int N)` which returns the $N^{th}$ element counting from the END of the list. Print the contents of the list as well as the $N^{th}$ to last element on separate lines.

7. (10pts) Implement a doubly-linked list. Write the function `reverse(struct listnode** head)` which reverses the list by swapping the previous and next pointers of each node. Print the contents of the list before and after the reversal.

8. (10pts) Using your doubly-linked list and reverse function, write a function `is_palyndrome(struct listnode* head)` that returns 1 if the string encoded in your list is a palyndrome and 0 if it is not.

9. (10pts) Construct a binary search tree (BST) that stores integers (i.e. the descendants to the left of a node are less than or equal to the node and the descendants to the right of the node are greater than or equal to the node). Insert integers 100, 75, 50, 125, 25, 150 into the tree. Write the function `inorder_print(treenode*)` that uses recursion to perform a inorder traversal of the tree and print the values.

10. (10pts) See previous problem. Write the function `nr_inorder_print(treenode*)` that performs a inorder traversal of the tree WITHOUT recursion. Instead, implement a stack (singly-linked list manipulated with push(...)/pop(...) functions) to aid with the traversal.

# 3 What to Turn In

Via Moodle, individually turn in a single compressed tarball, containing exactly the directory `hw0` and its contents, named `hw0.tgz`.