James Snee s3369721 Algorithms and Analysis Summer Semester 2015 Assignment 2

Problem 01: Formal Language for Simple Algebra

2. The Data Structure we will use for evaluating correct parentheses is a Stack. Whenever an open bracket (occurs, it gets pushed to the stack, and when a closing bracket) occurs, the open bracket at the top gets popped off the top of the stack (note that no closing brackets are pushed to the stack as there is no need). Once the stack is empty, we have the correct number of parentheses. Similarly, if the stack contains open brackets, that means that there are still some open brackets that haven't been closed, therefore the string is not accepted. Using the example string:

((a) + (b)) + ((c)(a))

The stack will contain nothing at the end, as there are the correct number of opens and closes (12, 6 opens and 6 closes). Note that there needs to be an even number of opens and closes, for every open there must be a close. Please refer to the code attached. PLEASE NOTE: The code uses portions of the Stack Solution released on blackboard.

The functions/operations that we will use:

void stack_init(stack_s *stack, size_t size); //Initialises stack data structure, allocating memory as necessary.

void stack_push(stack_s *stack, char item); //pushes an item to the stack, in our case the char '('.

char stack_pop(stack_s *stack); // pops a char from the top of the stack.

int stack_is_empty(stack_s *stack); // checks to see if the stack is empty at the end. Returns 1 if empty, 0 if not.

void stack_free(stack_s *stack); // frees the allocated memory to the stack structure.

Problem 02: Hash Table

0	1	2	3	4	5	6	7	8	9	10	11
12			11	16	17	28				22	35

Hash function: H(k) = k mod 12. Hash function 2: H2(k) = 7 – (k mod 7)
Collision occurred with values 16 and 28, resolved with Hash function 2, which gives us a value of 7, so we jump 7 spaces (11 is already taken so we jump another 7) to put 28 into position 6 of the array.