



Data Structures and Program Design Using C

SCS 1301

AY 21 - Semester 1



Exercise VII

Dr Manjusri Ishwara

1. Following exercises are to be completed using a stack data structure. You must implement the stack data structure so that types and functionality are in separate code files.
 - (a) Implement a stack data structure that can be used to insert integers.
 - (b) Number addition can be performed with the use of stacks. Using the stack developed in part (a) above, write a program to perform number addition using the algorithm below. Note that the user has to provide the two operands for the addition.
 - i. Initialize three (3) stacks, two (2) stacks for the two operands and one stack for the results.
 - ii. Each operand is to be pushed into their respective stack one digit at a time.
 - iii. Pop both stacks and add the result. Push the result into the result stack. If the addition is or exceeds ten (10), carry one forward to the next addition by step (ii) above and push the 10^0 position to the result stack.
 - iv. Repeat steps (ii) and (iii) until both operand stacks are empty.
 - v. Pop the digits from the result stack to get the final answer.
 - (c) Extend the stack in part (a) to accept character values.
 - (d) Using the stack implemented in part (c) above, implement a delimiter and a bracket matcher when a character string of any length is given as user input. The algorithm is given below.
 - i. The following brackets are to be considered in this program.
 - Square Brackets [...]
 - Curly Brackets {...}
 - Parentheses (...)
 - ii. Read the input string character by character, and if it matches any of the brackets above, process according to step (iii) below. If another character is read, ignore the character and read the next character.
 - iii. If the read character is an opening symbol from the list above, push the symbol into the stack. If the read character is a closing symbol, pop the stack and compare if the popped character is the matching opening symbol. If it is a match, read the next character. Print an error saying symbols do not match and end the program in case the brackets do not match.
 - iv. The process stops when the entire input string is processed, and the stack is empty. A message saying that all brackets are in order should be displayed in such cases.

2. Following exercises are to be completed using a queue data structure. You must implement the queue data structure so that types and functionality are in separate code files.
- (a) Implement a priority queue with three priority levels: ‘High’, ‘Medium’, and ‘Low’. The tasks with different priorities come to the queue at random. The dequeue will be based on the height priority.
- i. Use three queues for the three priorities.
 - ii. Use a single queue to store all three priorities.



UCSC