Name: Shuhao Zhang ID:201138949 x5sz2

COMP281 Assignment 3 Report

The problem 1046 is to compute the sum of adjacent numbers in the n x n grid. m is the number of the adjacent number that should be summed. In addition, any direction (up, down, left, or diagonally) in the grid should be considered. In my program, first of all, input the integer n and m to allocate the memory for n x n grid. Then input the integer to the matrix one by one. Next, input the matrix, integer n and integer m to the method called “result_largest”, and the method return an integer which is the answer. Finally, print out the answer and free the memory of matrix. In the method “result_largest”, there are five variates to store the value of sum in row, sum in column, sum in 2 diagonal directions and maximum of all the sum. There are three nested “for” loop to compute the sum. The first “for” loop and second “for” loop is used to set the position of the start point in the grid. The start point means the first number of m adjacent numbers. Then the innermost “for” loop is to move the pointer from the start point to the next adjacent number in one direction and compute the sum of (at most)m number. In addition, there are some “if” command to avoid the pointer move outside the matrix. At last, each sum in different direction compares with the maximum and update the maximum. Finally, the method return the maximum.

The problem 1048 is to sort and search a given list of dates. The input data should be one string, one integer (between 1-31) and one two digit integer which indicates month, day and year. To sort the date and search the specific date, the method qsort and bsearch have been used in my code. The method called “cmp” is used to compare the time of two date by year, month and day. The method “countMonth” is used for transfer the string of month to the number. The date is defined as structure by a string and three integers which store number of month, number of day and number of year. Firstly, input the number of dates and allocate the memory for these dates. After user inputs the date, and also the program get the query. Using the qsort to sort the list of dates from latest date to oldest date and using the bsearch to search the answer of query. Finally, the sorted list of dates and the answer of query could be printed out. After the answer is printed, free all the memory.

The problem 1053 is to implement the stack structure in C. The stack follows last in, first out (LIFO) rule and contains two fundamental operations called push and pop. To implement the stack structure, the node structure should be defined. The node structure contains a pointer to the next node and an integer to store the value. For the main method, there are two string s1, s2 which store the words of two operations (Push and Pop). There is a “while” loop to continuous receive user's input until the EOF. There are two "if" commands to recognize two operations. For the method "push_node", first of all, allocate the memory for a date and then add the value from input to the node. The new node always add to the head of this linked list. For the method "pop_node", if the list is empty, the method return -1. If the list is not empty, the method return the value of the head of the list and the head of this list will be removed. For every operations, a sentence about operations is printed. After all the operations finished, all the memory will be freed.

The problem 1049 is to implement the priority queue in C. The priority queue can be implemented using linked lists. The priority queue arrange the elements from high priority to low priority. First of all, the node structure should be defined. It contains an integer to store value, an integer to store priority and a pointer to the next node. For the main method, there are two string s1, s2, which store the words of two operations (Insert and Pop). The "While" loop can continuous receive user's input until the EOF. There are two "if" commands to recognize these two operations. For the method "insert_node", the program allocates the memory for a node for storing the value and priority. Then, this node is added to the suitable position of the list based on its priority. For the method "pop_node", the program can return the value of the head of the list, which is the highest priority element. Then, the node in head of the list will be removed. If the list is empty, then this method will return -1. For every method "pop_node", the value of the removed node will be printed out. Finally, after all the operations finished, all the memory will be freed.

The problem 1050 is to find if there is route between a given pair of cities. To solve this problem, I try to use the breadth-first search to find the solution. First, the node structure contains an integer to store the id of city and a pointer that points to the next node. After the user input the number of cities and lines, the program will allocate the memory to create the same number nodes to indicate the city, so there will be m (m is the number of city) linked list. Then when the user input one pair of number (line), the program will create a new node which the number is the same as the larger number in the pair. Then the new node will be added to a linked list which the number of head node is the same as the smaller number in the pair. After the program get all the links and the query, the method "search_link" will compute the answer and return back. For the method "search_link", the program will search the linked list which the first node corresponds to the first number of pair. If the program cannot find the second number of the pair in this linked list. Then the program starts to find the linked list which corresponds to the number of second node. In this way, if the program cannot find the answer after it search the last node, the method will return 0. If the program find the answer, it will return 1. Finally, print out the answer and free all the memory.