LINKED LIST – STACK – QUEUE EXERCISES

Linked List Exercises

Problem 1. Assume that a circular doubly linked list has been created, as in Figure 1. After each of the following assignments, indicate changes made in the list by showing which links have been modified. The second assignment should make changes in the list modified by the first assignment, and so on.

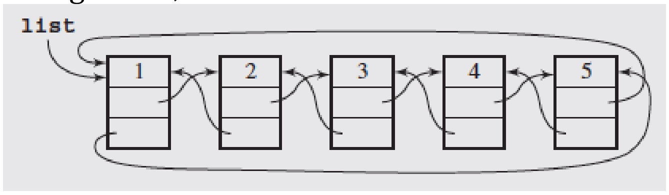

Fig 1. A circular doubly linked list

```
a. list->next->next->next = list->prev;
b. list->prev->prev->prev = list->next->next->next->prev;
c. list->next->next->next->prev = list->prev->prev->prev;
d. list->next = list->next->next;
e. list->next->prev->next = list->next->next->next;
```

Problem 2. Given a singly linked list *L*, each node contains an integer as key and a pointer to its successor. Create two new singly linked lists: *L1* contains even numbers of the given linked list and *L2* contains odd numbers of the list. Print out the two lists.

Example: *L* = <1, 5, 6, 8, 4, 3, 2>
L1 = <1, 5, 3>
L2 = <6, 8, 4, 2>

Problem 3. Given a singly linked list L, write functions for the following tasks:

a. Count the number of nodes in *L*
b. Search for the *i*th node in *L*, return the address of that node if found
c. Insert a node x after a node *k* in *L*
d. Delete the node before the node *k* in *L*
e. Reverse *L*

Problem 4. Write a function to check whether two singly linked lists have the same contents.

Stack Exercises

Problem 1. Given the struct stack as follows. Implement all the operations using C++. For the functions which has an integer return type, return 0 if the operation fails, return 1 if it succeeds.

```
struct video {
    char* title;
    char category[5];
    double score;
};

struct node {
    video data;
    node* pNext;
};

struct stack {
    node* pTop;   //pointer that points to the top of the stack
};
```

a. `bool IsStackEmpty(stack s);`
b. `int Count(stack s);     //count the number of elements in stack`
c. `int Push(stack& s, const video& data);`
d. `int Pop(stack& s, video& data);`
e. `int Top(stack s, video& data);`
f. `int DeleteStack(stack& s); //delete all elements in stack`
g. `int RemoveTop(stack& s, int n); //Remove topmost n entries from a stack`
h. `int DisplayAll(stack s); //display all elements in stack`

Problem 2. Display the content of a stack S in reverse order; that is, display the top last.

Problem 3. Delete every occurrence of a specified item from stack S, leaving the order of the remaining items unchanged.
E.g., $S = 10, 9, 8, 2, 5, 10, 7, 3, 10, 12$ → After delete 10: $S = 9, 8, 2, 5, 7, 3, 12$

Problem 4. Reverse the order of elements on stack S using:
a. Two additional stacks
b. One additional queue
c. One additional stacks and some additional nonarray variables

Problem 5. Put the integers on the stack S in ascending order using one additional stack and some additional nonarray variables.

Problem 6. Transfer elements from stack S_1 to stack S_2 so that the elements from S_2 are in the same order as on S_1
a. using one additional stack
b. using no additional stack but only some additional nonarray variables

Problem 7. Show the contain of the stack S after each operations in the following sequence:

EAS*Y**QUE***ST***I*ON

Each letter demonstrates a push operator pushing that letter into S, the "*" demonstrates a pop operator which remove the top element of S and print it out. What are printed after we finish reading the above sequence?

Problem 8. Show the contain of S and the values of each variable $A = 5, B = 3, C = 7$ after the following operations:

a. Initialize a new stack S
b. Push A to S
c. Push $C \times C$ to S
d. Pop and save the result to B
e. Push $B + A$ to S
f. Pop and save the result to A
g. Pop and save the result to C

Problem 9. Evaluate the following postfix expressions by using stack. Show the status of the stack after each step. Assume the following values for the identifiers: $a = 7$, $b = 3, c = 12, d =-5, e = 1$.

a. $a\ b\ c + -$
b. $a\ b\ c - d\ \times$
c. $a\ b + c - d\ e\ \times +$

Problem 10. Convert the following infix expressions to postfix form by using the algorithm given in the slide. Show the status of the stack after each step of the algorithm

a. $a - b + c$
b. $a - (b \div c \times d)$
c. $a \div (b \times c)$
d. $a \div b \div c - (d + e) \times f$
e. $(a + b) \times c$
f. $a \times (b \div c \div d) + e$
g. $a - (b + c)$
h. $a - (b + c \times d) \div e$

Problem 11. Suppose that an intermixed sequence of push and pop operations are performed. The pushes push the integers 0 through 9 in order; the pops print out the return value. Which of the following sequences could not occur?

(a) 4 3 2 1 0 9 8 7 6 5
(b) 4 6 8 7 5 3 2 9 0 1
(c) 2 5 6 7 4 8 9 3 1 0
(d) 4 3 2 1 0 5 6 7 8 9

Problem 12. Write a program that reads in a positive integer and prints the binary representation of that integer.

Problem 13. A palindrome is a word, number, phrase, or other sequence of symbols that reads the same backwards as forwards, such as the words *madam* or *racecar*. Write a function `IsPalindrome` which uses a stack to check if a string is a palindrome or not.

Queue Exercises

Problem 1. Given the struct queue as follows. Implement all the operations using C++. For the functions which has an integer return type, return 0 if the operation fails, return 1 if it succeeds.

```
struct video {
    char* title;
    char category[5];
    double score;
};

struct node {
    video data;
    node* pNext;
};

struct queue {
    node* pFront; //pointer to the 1st element
    node* pRear;  //pointer to the last element
};
```

a. bool IsQueueEmpty(queue q);
b. int Count(queue q); //count the number of elements in queue
c. int Enqueue(queue& q, const video& data);
d. int Dequeue(queue& q, video& data); //delete the 1st element in queue
e. int First(queue q, video& data); //get the 1st element in queue
f. int DeleteQueue (queue& q); //delete all elements in queue
g. int RemoveFirst(queue& q, int n); //Remove first n entries from a queue
h. int DisplayAll(queue q); //display all elements in queue

Problem 2. Use array to implement a queue. Do the questions 1a to 1h again. Include 1 more function IsQueueFull to check if the array reached the max size or not. The array is circular.

```
struct queue {
    video* arr;     //array of video
    int n;          //max size of the array
    int front;      //index of the 1st element
    int tail;       //index of the last element
};
```

Problem 3. Write a link-based implementation of a queue that uses a circular linked list to represent the items in the queue. You will need a single *start* pointer. When you are done, compare your implementation to the one in Problem 1. that uses a linear linked list with two external pointers. Which implementation is easier to write? Which is easier to understand? Which is more efficient?

Problem 4. Show how to implement a queue using 2 stacks?