Experiment set 2(b)

Stack, Queue, Expression Evaluation

Ex1. Implement the stack ADT for string data as we talked about in class. You should use linked list-based representation. Maximum stack size is unlimited. Assume NO global pointer and pass all required variables as parameter to function. Your code should at least satisfy the following interface:

```
 struct stack

  typedef struct stack *stack;

  stack newStack ();
  void stackPush (stack* stk, string x);
  string stackPop (stack* stk);
  int stackSize (stack stk);
  int stackIsEmpty (stack stk);
  void stackTop (stack stk);   [print top ]
```

Test case to evaluate your stack implementation:

1. Create stack
2. Push <the latest movie released you want to watch>
3. Pop
4. Pop (check if stack is empty then show the error message that “stack empty”)
5. Push three movies name you want to watch next.
6. Pop
7. Push 2 more movies name in your stack
8. What is the length of stack now
9. Print the list of movies are now in your stack with help of stackTop() and stackPop() .

Homework

Ex2. Implement the circular queue ADT for structure for integer data as we talked about in class.You should use circular queue implementation. Queue should

accommodate only 10 integers. Your code should at least satisfy the following interface:

```
  typedef struct queue *queue;

  queue newQueue ();
  void enQueue (queue q, int x);
  poly deQueue (queue q);
  int queueSize (queue q);
  int queueIsEmpty (queue q);

  int queuesize (queue q);
 void queueFront ((queue q);   [print front of the queue ]
```

1. Create Queue
2. enQueue 1 2
3. deQueue
4. deQueue
5. deQueue (show error message if empty)
6. enQueue 1 2 3 4 5
7. deQueue
8. Print the queue size.
9. enQueue 1 2 3 4 5 6 7 8 9 10 (show error when queue is empty)
10. Print all number in queue. Use deQueue() and queueFront()

Ex3: C program to evaluate value of a postfix expression.

Test case : Evaluate 231*+9- . The answer is -4

Following is algorithm for evaluating postfix expressions.

1) Create a stack to store operands (or values).
2) Scan the given expression and do following for every scanned element.
…..a) If the element is a number, push it into the stack
…..b) If the element is a operator, pop operands(two) for the operator from stack. Evaluate the operator and push the result back to the stack
3) When the expression is ended, the number in the stack is the final answer.

Ex4 C program to convert infix to postfix expression

1. Print operands as they arrive.
2. If the stack is empty or contains a left parenthesis on top, push the incoming operator onto the stack.
3. If the incoming symbol is a left parenthesis, push it on the stack.
4. If the incoming symbol is a right parenthesis, pop the stack and print the operators until you see a left parenthesis. Discard the pair of parentheses.
5. If the incoming symbol has higher precedence than the top of the stack, push it on the stack.
6. If the incoming symbol has equal precedence with the top of the stack, use association. If the association is left to right, pop and print the top of the stack and then push the incoming operator. If the association is right to left, push the incoming operator.
7. If the incoming symbol has lower precedence than the symbol on the top of the stack, pop the stack and print the top operator. Then test the incoming operator against the new top of stack.
8. At the end of the expression, pop and print all operators on the stack. (No parentheses should remain.)

Test case: convert (2+3*1)-9 to 231*+9-