Name: Mike Mico
Std num: 3685120

Q2.

Complete the programs used in lab adding any necessary features. Remember the integer num in an S2Node is to be changed to a double and all appropriate later changes. Add a major function (separate file) which accepts a queue of S2Nodes which represents an in-fix expression and have it convert to a queue of post fixed expressions based on the rules discussed in class (and on the hand out). You will need a function to print out a queue to a single line in the output. Show the print out of your Queue before conversion and after. Your program does not have to calculate the answer to either expression. Hand in a make file for compiling this program too

```
#ifndef S2NODE
#define S2NODE
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
/* ********************************************************
Program:     Queue.h
Purpose:    To define a queue of S2Nodes which contain either
            an integer or a character value
Date:        Programmed on November 17th, 2021
Author:             Programmed by Dr. Jeffrey Mark McNally, revised by Mike Mico
******************************************************** */


/* ********************************************************
Structure Definition for Snode
          A Snode of this type can hold either an integer or a character
          We use a boolean variable (isNum) to say which but have space for both
                  isNum == true  means this node contains an integer
                  isNum == false means this node contains a characer
                                              which can only be +, -, *, /, ^, (, or )
               If the contents are a character, we assign a precedence according to
                           +    1
                           -    1
                           *    2
                           /    2
                           ^    3
                           (    4
                           )    4
******************************************************** */
typedef struct S2Node
{
    double num;
    char symbol;
    int precedence;
    bool isNum;
       struct S2Node* next;
```

```
}s2Node;



/* ********************************************************
Program:     Snode.h
Purpose:    To define a Singly Linked Node which contains either
            an double or a character value
Date:        Programmed on November 9th, 2019
Author:            Programmed by Dr. Jeffrey Mark McNally
********************************************************** */
s2Node* createS2Node(double inNum, char inChar, bool isNum)
{
    s2Node* newNode = (s2Node*) malloc(sizeof(s2Node));
    newNode->isNum = isNum;
      //printf("creating a node with a %s \n", (newNode->isNum) ? "number":
"character");
      newNode->num = inNum;
    newNode->symbol = inChar;
    //printf("num = %lf  and char is %c \n", newNode->num, newNode->symbol);

    if (isNum)
      {
       newNode->isNum = isNum;
       //printf("making a node with double % \n", newNode->num);
      }
      else
    {
       //printf("making a node with character %c \n", newNode->symbol);
            switch (newNode->symbol)
            {
                  case '+':
                  case '-':
                        newNode->precedence  = 1;
                        break;
                  case '*':
                  case '/':
                        newNode->precedence  = 2;
                        break;
                  case '^':
                        newNode->precedence  = 3;
                        break;
                  default:
                        newNode->precedence  = 4;
            }
      }
    newNode->next = NULL;
    return newNode;
}



void printS2Node(s2Node* current)
{
      printf("Boolean is %s:  contains ", (current->isNum) ? "true": "false");
```

```
    if (current->isNum)
          printf("%lf", current->num);
       else
       printf("%c",  current->symbol);
    printf(" at %9d with a next at %9d \n\n", current, current->next);
}

void destroy(s2Node* current)
{
    current->num = rand();
    current->symbol = rand();
    current->next = NULL;
    free(current);
}



#endif
```

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#ifndef STACK
#define STACK

#include "S2Node.h"
/* ***********************************************************
Program:     Queue.h
Purpose:    To define a stack of S2Nodes which contain either
            an integer or a character value
Date:        Programmed on November 17th, 2021
Author:             Programmed by Dr. Jeffrey Mark McNally, revised by Mike Mico
************************************************************* */


/* ***********************************************************
Structure Definition for Stack
          A stack  contains a top and nothing else.
               It is a Last in, first out structure
************************************************************** */
typedef struct Stack
{
       s2Node* top;

}stack;


stack* createStack()
{
       stack* newStack = (stack*) malloc( sizeof(stack) );
       newStack->top = NULL;

       return newStack;
}

/*************************************************************
declare methods
*************************************************************/

void pushDouble(stack* myStack, double num);
void pushChar(stack* myStack, char myChar);
double popDouble(stack* myStack);
char popChar(stack* myStack);
bool isEmptyStack(stack* myStack);
bool isTopDouble(stack* myStack);
bool isTopChar(stack* myStack);
double peekDouble(stack* myStack);
char peekChar(stack* myStack);

//push a number to the stack
void pushDouble(stack* myStack, double num)
{
    //initialie variables
    s2Node* newnode;
```

```
    s2Node* current;
    current = myStack->top;

    //set isNum boolean to true showing it is a number
    newnode = createS2Node(num,'_',true);
    //condition 1- empty stack
    //make the top our new node
    if(isEmptyStack(myStack))
        myStack->top = newnode;
    //condition 2- not empty staack
    //make the top our new node
    //make it now point to the previous top
    else
    {
        myStack->top = newnode;
        newnode->next = current;
    }
 }

 //push a character to the stack
void pushChar(stack* myStack, char myChar)
{
    //initialize variables
    s2Node* newnode;
    s2Node* current;
    current = myStack->top;

    //create char and macke sure bool isNum is false
    newnode = createS2Node(0.0,myChar,false);
    //condition 1 - empty stack
    //condition 2- not empty staack
    //make the top our new node
    //make it now point to the previous top
    if(isEmptyStack(myStack))
        myStack->top = newnode;
    else
    {
        myStack->top = newnode;
        newnode->next = current;
    }
}

//pop a double from the top of the stack
double popDouble(stack* myStack)
{
    //initialize variables
    s2Node* current;
    current = myStack->top;
    double value;
    //condition 1 - empty stack
    if(isEmptyStack(myStack))
    {
        printf("stack is empty \n");
    }
    // condition 2 - not empty stack
```

```
    // store the number we are popping in the variable 'value'
    // make the next value the top
    // destroy the current top
    // return our variable 'value'
    else
    {
        if (current->isNum)
        {
            value = current->num;
            myStack->top = current->next;
            destroy(current);
            return value;
        }
    }
}


char popChar(stack* myStack)
{
    s2Node* current;
    current = myStack->top;
    char thisChar;
    // condition 1 - stack is empty
    if(isEmptyStack(myStack))
    {
        printf("stack is empty \n");
    }
    // condition 2 - not empty stack
    // store the number we are popping in the variable 'thisChar'
    // make the next value the top
    // destroy the current top
    // return our variable 'thisChar'
    else
    {
        if (!current->isNum)
        {
            thisChar = current->symbol;
            myStack->top = current->next;
            destroy(current);
            return thisChar;
        }
    }
}

//check if stack is empty
bool isEmptyStack(stack* myStack)
{
    //if the stack is eempty return true
    //otherwise return false
    if(myStack->top == NULL)
        return true;
    else
        return false;
}
```

```
//check if the top is an int
bool isTopDouble(stack* myStack)
{
    //if the top is a double return true
    //otherwise return false
    if (myStack->top->isNum)
        return true;
    else
        return false;

}

//check if the top is a char
bool isTopChar(stack* myStack)
{
    //if the top is a character return true
    //otherwise return false
    if (!myStack->top->isNum)
        return true;
    else
        return false;
}

//return the top if its a double but do not remove it
double peekDouble(stack* myStack)
{
    //condition 1 - empty stack
    if(isEmptyStack(myStack))
    {
        printf("stack is empty \n");
    }
    //condition 2 - not empty
    //if the top is a double return that double
    else
    {
        if (isTopDouble(myStack))
        {
            return myStack->top->num;
        }
    }
}

//return the top if its a character but do not remove it
char peekChar(stack* myStack)
{
    //condition 1 - empty stack
    if(isEmptyStack(myStack))
    {
        printf("stack is empty \n");
    }
    //condition 2 - not empty
    //if the top is a character return that character
    else
    {
        if (isTopDouble(myStack))
```

```
        {
            return myStack->top->num;
        }
    }
}

void printStack(stack* myStack)
{
       //initialize variables
       s2Node* Current;
       Current = myStack->top;

       //condition 1 - staack is empty
       if (isEmptyStack(myStack))
              printf("stack is empty \n");
    //condition 2 - not empty
       else
       {
           //loop through the stack starting at the top
              while(Current != NULL )
              {
                  //check whether current is double or char then print accordingly;
                     if (Current->isNum)
                            printf("%.2lf ",Current->num);
                     else
                            printf("%c ",Current->symbol);
                     Current = Current->next;
              }
       }
       printf("\n");

}
#endif
```

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#ifndef QUEUE
#define QUEUE

#include "S2Node.h"
#include "Stack.h"
/* ************************************************************
Program:     Queue.h
Purpose:    To define a queue of S2Nodes which contain either
            an integer or a character value
Date:        Programmed on November 17th, 2021
Author:             Programmed by Dr. Jeffrey Mark McNally, revised by Mike Mico
************************************************************* */

/* ************************************************************
Structure Definition for Queue
          A stack  contains a head and a tail.
               It is a First in, first out structure
************************************************************** */
typedef struct Queue
{
       s2Node* head;
       s2Node* tail;

}queue;


queue* createQueue()
{
       queue* newQueue = (queue*) malloc( sizeof(queue) );
       newQueue->head = NULL;
       newQueue->tail = NULL;

       return  newQueue;
}

//initialize methods
void enQueueDouble(queue* myQ, double num);
void enQueueChar(queue* myQ, char myChar);
double deQueueDouble(queue* myQ);
char deQueueChar(queue* myQ);
bool isEmptyQueue(queue* myQ);
bool isHeadDouble(queue* myQ);
bool isHeadChar(queue* myQ);

void enQueueDouble(queue* myQ, double num)
{
       //initialize variables
    s2Node* newnode;
    s2Node* current;
    current = myQ->tail;

    //make sure to set boolean value as true to show our newnode is an int
```

```
    //make a dummy char to fill the parameters
    newnode = createS2Node(num,'_',true);
    //condition 1 - if the queue is empty
    if(isEmptyQueue(myQ))
    {
        myQ->head = newnode;
        myQ->tail = newnode;
    }
    //any other condition
    else
    {
        current->next = newnode;
        myQ->tail = newnode;
    }
}

void enQueueChar(queue* myQ, char myChar)
{
      //initialize variables
    s2Node* newnode;
    s2Node* current;
    current = myQ->tail;

    //set boolean value to false and num value can be a dummy like 0
    newnode = createS2Node(0.0,myChar,false);
    //condition 1- if queue is empty
    if(isEmptyQueue(myQ))
    {
        myQ->head = newnode;
        myQ->tail = newnode;
    }
    //other conditions
    else
    {
        current->next = newnode;
        myQ->tail = newnode;
    }
}

// this method is ONLY being called when the top node is a char

double deQueueDouble(queue* myQ)
{
      //initialize variablees
    s2Node* current = myQ->head;

    double value = current->num;
      myQ->head = current->next;
    destroy(current);
    return value;

}


// this method is ONLY being called when the top node is a char
```

```
char deQueueChar(queue* myQ)
{
       //initialize variables
    s2Node* current = myQ->head;

    char thisChar = current->symbol;
    myQ->head = current->next;
    destroy(current);
    return thisChar;

}

//method to check if the queue is empty
bool isEmptyQueue(queue* myQ)
{
    if (myQ->head ==NULL)
        return true;
    else
        return false;
}

//method to check if the head is a double
bool isHeadDouble(queue* myQ)
{
    if (myQ->head->isNum)
        return true;
    else
        return false;
}

//method to check if the head is a character
bool isHeadChar(queue* myQ)
{
    if (!myQ->head->isNum)
        return true;
    else
        return false;

}

//print the queue from head to tail
void printQ(queue* myQ)
{
       //initialize variables
       s2Node* curr;
       curr = myQ->head;

       //condition 1 - if queue is empty
       if (isEmptyQueue(myQ))
              printf("queue is empty \n");
    //queue is not empty
       else
       {
           //loop through the queue
```

```
            while(curr != NULL )
            {
                //check whether the current is a num or char then print accordingly
                  if (curr->isNum )
                         printf("%.2lf ",curr->num);
                  else
                 printf("%c ",curr->symbol);
                  curr = curr->next;
            }
      }
      printf("\n");
}

#endif
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "Queue.h"


/* ********************************************************
Program:     postfix.h
Purpose:    convert expressions from infix to postfix
Date:        Programmed on December 5th, 2021
Author:             Programmed by Mike Mico
******************************************************** */

queue* postfix(queue* myQ)
{
       //initialize variables
       stack* stck = createStack();
       queue* emptyQ = createQueue();
       s2Node* cur = myQ->head;
       bool closedBracket = false;

    //if we have an empty queue return it
       if(isEmptyQueue(myQ))
              return emptyQ;
    //loop through the queueu
       while(cur != NULL)
       {
              //is a number
              if(cur->isNum)
              {
                     cur = cur->next;
                     enQueueDouble(emptyQ,deQueueDouble(myQ));
              }

              //is a character
              else
              {
                     if(isEmptyStack(stck))
                     {
                            cur = cur->next;
                            pushChar(stck,deQueueChar(myQ));
                     }

                     else
                     {
                            if(cur->precedence  >  stck->top->precedence )
                            {
                                   //not brackets
                                   if(cur->precedence != 4)
                     {
                         cur = cur->next;
                         pushChar(stck,deQueueChar(myQ));
                     }
                                   //open bracket
```

```
                    else if(cur->precedence == 4)
                    {
                if(closedBracket == false)
                {
                    cur = cur->next;
                    pushChar(stck,deQueueChar(myQ));
                    stck->top->precedence = 0;
                    closedBracket = true;
                }
                else
                {
                    while (stck->top->precedence != 0)
                    {
                        enQueueChar(emptyQ,popChar(stck));
                    }
                    popChar(stck);
                    closedBracket = false;

                }
                    }
                    //close bracket
                }
                //precedence is less
                  else
               {
               while (stck->top != NULL)
               {
                    if(stck->top->symbol == '(' || stck->top->symbol == ')')
                        popChar(stck);
                    else
                        enQueueChar(emptyQ,popChar(stck));
               }
                  }
               }
           }

      }


      while (stck->top != NULL)
      {
            if(stck->top->symbol == '(' || stck->top->symbol == ')')
          popChar(stck);
       else
          enQueueChar(emptyQ,popChar(stck));
      }

    return emptyQ;

}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "Stack.h"
#include "Queue.h"
#include "S2Node.h"
#include "postfix.h"

#define pi 3.14

/* ********************************************************
Program:     driver.c
Purpose:    test postfix methods,queue methods and stack methods
Date:        Programmed on December 5th, 2021
Author:             Programmed by Mike Mico
********************************************************** */



int main(int argc, char *argv[]) {
/*
       s2Node* tester = createS2Node(42, '_', true);
       printS2Node(tester);
       //free(tester);

       s2Node* tester2 = createS2Node(0, 'a', false);
       printS2Node(tester2);

       stack* thisStack = createStack();

       pushDouble(thisStack,43);
       pushDouble(thisStack,23);
       pushDouble(thisStack,11);
       pushChar(thisStack,'+');
       pushChar(thisStack,'-');

       printf("we are popping %c \n",popChar(thisStack));
       printf("we are popping %c \n",popChar(thisStack));
       printf("we are popping %lf \n",popDouble(thisStack));
       printf("we are popping %lf \n",popDouble(thisStack));
       printf("we are popping %lf \n",popDouble(thisStack));
       printf("we are popping %lf \n",popDouble(thisStack));
*/

       queue* thisQ = createQueue();


       queue* newQ ;

// 1 A

       enQueueDouble(thisQ,3);
       enQueueChar(thisQ,'*');
    enQueueDouble(thisQ,4);
       enQueueChar(thisQ,'+');
```

```
        enQueueDouble(thisQ,5);
        enQueueChar(thisQ,'+');
        enQueueDouble(thisQ,2);
        enQueueChar(thisQ,'*');
        enQueueDouble(thisQ,6);
        enQueueChar(thisQ,'-');
        enQueueDouble(thisQ,3);

        printf("equation 1 is \n");

        printQ(thisQ);
    printf(" \n");

        newQ = postfix(thisQ);
        printf("postfix notation \n");
        printQ(newQ);
    printf(" \n");
        thisQ = createQueue();
//1 B

        enQueueDouble(thisQ,3);
        enQueueChar(thisQ,'*');
        enQueueChar(thisQ,'(');
    enQueueDouble(thisQ,4);
        enQueueChar(thisQ,'+');
        enQueueDouble(thisQ,5);
        enQueueChar(thisQ,')');
        enQueueChar(thisQ,'^');
        enQueueDouble(thisQ,2);
        enQueueChar(thisQ,'*');
        enQueueDouble(thisQ,6);
        enQueueChar(thisQ,'-');
        enQueueDouble(thisQ,3);

    printf("equation 2 is \n");

        printQ(thisQ);
    printf(" \n");
        newQ = postfix(thisQ);
        printf("postfix notation \n");
        printQ(newQ);


//      free(tester);
//      free(tester2);
        return 0;
}
```

```
# driver makefile
# by : Mike Mico

COMPILER = gcc
CFLAGS = -g -Wall -Wshadow



runMe: driver.o
	$(COMPILER) $(CFLAGS) driver.o -o runMe

driver.o: driver.c
	$(COMPILER) $(CFLAGS) -c driver.c
run: runMe
	./runMe
```