CIS023 Laboratory 3 Assignment Mockup

Exercise L3-1 Chapter 16 – Linked Lists

Submit only the header files (`nodeType.h`, `linkedList.h`, `orderedLinkedList.h` and `unorderedLinkedList.h`) and your documentation files. The header files must include all function definitions. Validate compilation of your header files with the following code. The test program will exercise all functions in the classes listed above.

```
#include <iostream>
#include <string>
#include "orderedLinkedList.h"
#include "unorderedLinkedList.h"

using namespace std;

int main()
{
    orderedLinkedList<int> list1;
    orderedLinkedList<string> list2;

    unorderedLinkedList<int> list3;
    unorderedLinkedList<string> list4;

    list1.initializeList();
    list1.isEmptyList();
    list1.print();
    list1.length();
    list1.destroyList();
    list1.insertFirst(123);
    list1.front();
    list1.back();
    list1.insertLast(456);
    list1.deleteNode(123);
    list1.search(456);
    list1.search(123);

    nodeType<float> node1;
    node1.getInfo();
    node1.setInfo(123.0);
    node1 = node1;

    return 0;
}
```

Exercise L3-2 Chapter 17 – Stacks & Queues

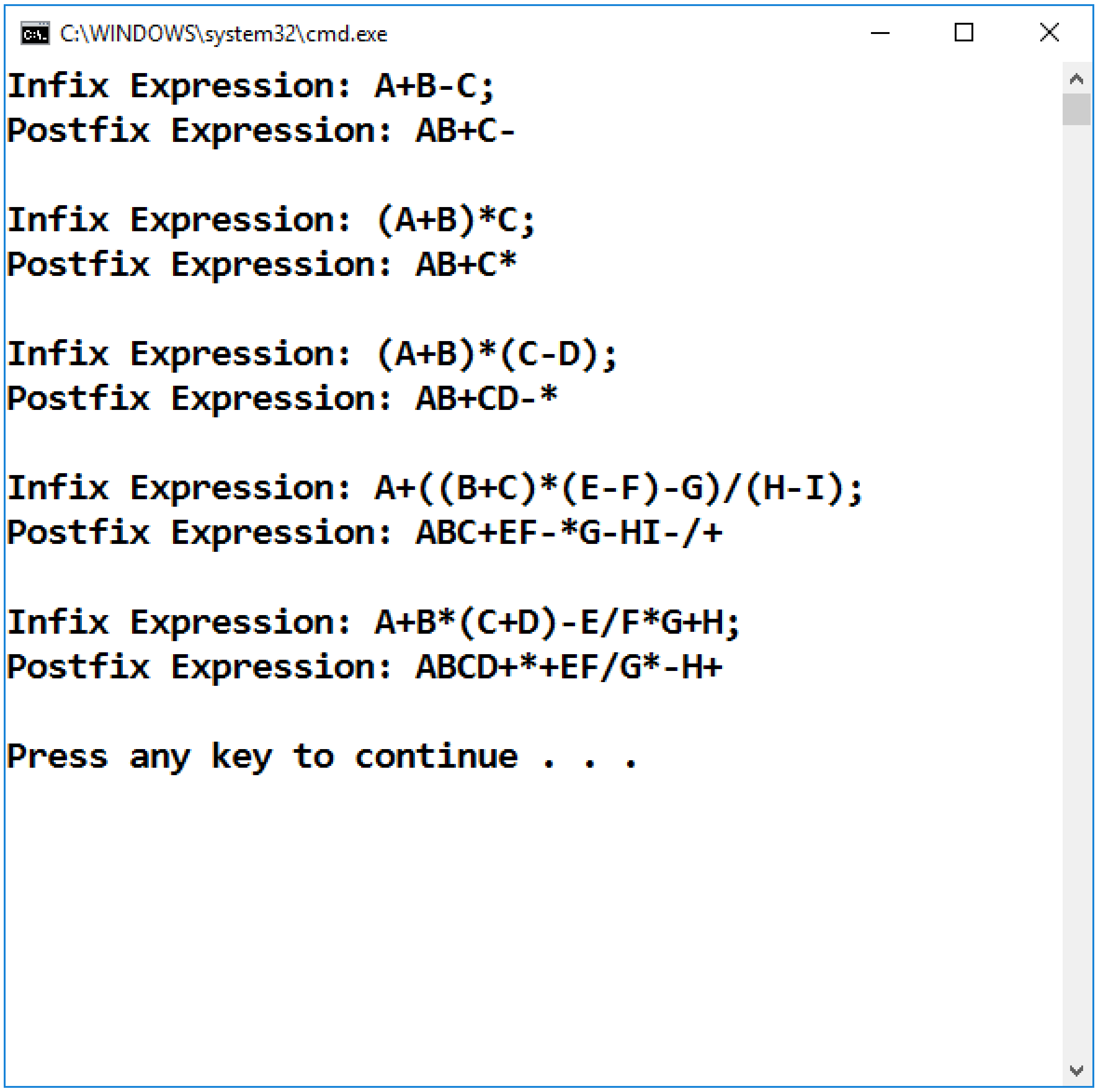
Submit only the header files (`myStack.h` and `infixToPostfix.h`) and your documentation files. The header files must include all function definitions. Validate compilation of your header files with the following code. The test program will exercise all functions in the classes listed above.

```
//Main program infix to postfix

#include <iostream>
#include <fstream>
#include <string>
#include "myStack.h"
#include "infixToPostfix.h"

using namespace std;

int main()
{
    infixToPostfix  InfixExp;
    string infix;
    ifstream infile;
    infile.open("infixData.txt");
    if (!infile)
    {
        cout << "Cannot open input file. Program terminates!!!" << endl;
        return 1;
    }
    getline(infile, infix);
    while (infile)
    {
        InfixExp.getInfix(infix);
        InfixExp.showInfix();
        InfixExp.showPostfix();
        cout << endl;
        getline(infile, infix);
    }
    infile.close();
    return 0;
     }
```



```
Infix Expression: A+B-C;
Postfix Expression: AB+C-

Infix Expression: (A+B)*C;
Postfix Expression: AB+C*

Infix Expression: (A+B)*(C-D);
Postfix Expression: AB+CD-*

Infix Expression: A+((B+C)*(E-F)-G)/(H-I);
Postfix Expression: ABC+EF-*G-HI-/+

Infix Expression: A+B*(C+D)-E/F*G+H;
Postfix Expression: ABCD+*+EF/G*-H+

Press any key to continue . . .
```

This page is Intentionally Blank for alignment

Exercise L3-3 Chapter 18 – Searching & Sorting Algorithms

Submit only the header file (`searchSortAlgorithms.h`) and your documentation files. The header file must include all function definitions. Validate compilation of your header files with the following code. The test program will exercise all functions in the classes listed above.

```
#include <iostream>
#include <ctime>
#include "searchSortAlgorithms.h"

using namespace std;

const int SIZE = 10000;

void fill(int list[], int size);
void copyList(int list[], int temp[], int length);

int main()
{
      int intlist[SIZE];
      int temp[SIZE];

      clock_t startTime1, endTime1;
      clock_t startTime2, endTime2;
      clock_t startTime3, endTime3;
      clock_t startTime4, endTime4;

      fill(intlist, SIZE);
      copyList(intlist, temp, SIZE);
      //Quick sort: The pivot is the middle element
      startTime1 = clock();
      quickSort(intlist, SIZE);
      endTime1 = clock();
      //Quick sort: The pivot is the median element
      copyList(temp, intlist, SIZE);
      startTime2 = clock();
      quickSortMedianPivot(intlist, SIZE);
      endTime2 = clock();
      //Quick sort with insertion sort
      //The pivot is the middle element
      copyList(temp, intlist, SIZE);
      startTime3 = clock();
      quickSortWithInsertionSort(intlist, SIZE);
      endTime3 = clock();
      //Quick sort with insertion sort
      //The pivot is the median element
      copyList(temp, intlist, SIZE);
      startTime4 = clock();
      quickSortMedianWithInsertionSort(intlist, SIZE);
      endTime4 = clock();
      cout << "Quick sort time, with pivot middle element: "
```

```
              << endTime1 - startTime1 << endl;
       cout << "Quick sort time, with pivot median element: "
              << endTime2 - startTime2 << endl;
       cout << "Quick sort and insertion sort time, with pivot "
              << "middle element: " << endTime3 - startTime3 << endl;
       cout << "Quick sort and insertion sort time, with pivot "
              << "median element: " << endTime4 - startTime4 << endl;

       return 0;
}



void fill(int list[], int size)
{
       int seed = 47;

       int multiplier = 2743;

       int addOn = 5923;

    int index = 0;

       while (index < size)
       {
          list[index++] = seed;
          seed = int(seed * multiplier + addOn);
       }

       cout << endl;
}

void copyList(int list[], int temp[], int length)
{
    for (int i = 0; i < length; i++)
        temp[i] = list[i];
}
```