Cover Page

Name: David Chen Salas

Section: 2023 Fall Term (1) Algorithms I CSCI 700 231[25504] (Queens College)

Project#: Project1(C++)

Project Name: Linked-list implementation of Stacks, Queues, and order Lists

Due Date: 9/13/2023 Wednesday before midnight

Algorithm Steps:

```
Step 0: inFile  open input file from argv[1]
        outFile1, outFile2, outFile3, deBugFile  open from argv[2], argv[3], argv[4], argv[5]
Step 1: S  creates a LLStack using constructor.
Step 2: buildStack (S, inFile, deBugFile)
Step 3: Q  creates a LLQueue using constructor.
Step 4: buildQueue (S, Q, outFile1, deBugFile)
Step 5: LL  creates a LLlist using constructor
Step 6: buildList (Q, LL, outFile2, deBugFile)
Step 7: printList (LL.listHead, outFile3)
Step 8: close all files
```

Source Code

```
#include <iostream>
#include <fstream>

using namespace std;

class listNode
{
public:
   int data;
   listNode* next;

   listNode(int data)
   {
      this->data = data;
      this->next = NULL;
   }

};
void printNode(listNode node);
void printNode(listNode node, ofstream& outFile);

class LLStack
{
public:
         listNode* top;

   LLStack()
   {
      top = new listNode(-9999);
   }

   void push(listNode newNode)
   {
      listNode* tmp = new listNode(newNode.data);
      tmp->next = top->next;
      top->next = tmp;
   }

   bool isEmpty()
   {
            if(top->next==NULL)
            {
         return true;
            }
```

```
        return false;
    }

    listNode* pop()
    {
            if(!this->isEmpty())
            {
          listNode* tmp = top->next;
          top->next = tmp->next;
          tmp->next = NULL;
          return tmp;
            }
        cout << "stack is empty";
        return nullptr;
    }
};
void buildStack(LLStack myStack, ifstream &inFile, ofstream &deBugFile);

class LLQueue
{
public:
    listNode* head;
    listNode* tail;

    LLQueue()
    {
        head = new listNode(-9999);
        tail = head;
    }

    void insertQ(listNode newNode)
    {
        listNode* tmp = new listNode(newNode.data);
        tail->next = tmp;
        tail = tmp;
    }

    bool isEmpty()
    {
        if (tail == head) {
            return true;
        }
            return false;
    }

};
```

```
void buildQueue(LLStack S, LLQueue& Q, ofstream& outFile1, ofstream& deBugFile);
listNode* deleteQ(LLQueue Q, ofstream& outFile2);

class LLlist
{
public:
   listNode* listHead;

   LLlist()
   {
      listHead = new listNode(-9999);
   }
};
listNode* findSpot(LLlist LL, listNode* newNode);
void insertOneNode(listNode* spot, listNode* newNode);
void buildList(LLQueue& Q, LLlist LL, ofstream& outFile2, ofstream& deBugFile);
void printList(listNode* listHead, ofstream& outFile3);

ifstream inFile;
ofstream outFile1;
ofstream outFile2;
ofstream outFile3;
ofstream deBugFile;
LLStack S;
LLQueue Q;
LLlist LL;

int main(int argc, const char* argv[])
{
         inFile.open(argv[1]);
   outFile1.open(argv[2]);
   outFile2.open(argv[3]);
   outFile3.open(argv[4]);
   deBugFile.open(argv[5]);

   S = LLStack();
         buildStack(S, inFile, deBugFile);

   Q = LLQueue();
   buildQueue(S, Q, outFile1, deBugFile);

   LL = LLlist();
   buildList(Q, LL, outFile2, deBugFile);
   printList(LL.listHead, outFile3);

   inFile.close();
```

```
    outFile1.close();
    outFile2.close();
    outFile3.close();
    deBugFile.close();
    return 0;
}

void printNode(listNode node)
{
    cout << "(" << node.data << ", " << node.next->data << ") -> ";
}

void printNode(listNode node, ofstream& outFile)
{
    outFile << "(" << node.data << ", " << node.next->data << ") -> ";
}

void printList(listNode* listHead, ofstream& outFile3)
{
    outFile3 << "listHead -> ";
    listNode* tmp = listHead;
    while (tmp->next != NULL)
    {
        printNode(*tmp, outFile3);
        tmp = tmp->next;
    }
    outFile3 << "NULL";
}

listNode* deleteQ(LLQueue Q, ofstream& outFile2)
{
    deBugFile << "entering deleteQ method!" << endl;
            listNode* tmp = Q.head->next;
    Q.head->next = tmp->next;
    tmp->next = NULL;
    deBugFile << tmp->data << endl;
    deBugFile << "leaving deleteQ method!" << endl;
    return tmp;
}

void buildStack(LLStack S, ifstream &inFile, ofstream &deBugFile){

            int data;
    deBugFile << "entering buildStack!" << endl;
    while (inFile >> data) {
        deBugFile << data << endl;
        listNode newNode = listNode(data);
        S.push(newNode);
    }
    deBugFile << "leaving buildStack!" << endl;
}

void buildQueue(LLStack S, LLQueue& Q, ofstream& outFile1, ofstream& deBugFile)
{
    deBugFile << "entering buildQueue!" << endl;
    while (!S.isEmpty()) {
        listNode newNode = listNode(S.pop()->data);
        deBugFile << newNode.data << endl;
        outFile1 << newNode.data << endl;
        Q.insertQ(newNode);
    }
    deBugFile << "leaving buildQueue!" << endl;
}

void buildList(LLQueue& Q, LLlist LL, ofstream& outFile2, ofstream& deBugFile)
{
    deBugFile << "entering buildList!" << endl;
    while (Q.head->next != NULL) {
        listNode* newNode = deleteQ(Q, outFile2);
        outFile2 << newNode->data << endl;
        deBugFile << newNode->data << endl;
        listNode* spot = findSpot(LL, newNode);
        insertOneNode(spot, newNode);
    }
    deBugFile << "leaving buildList!" << endl;
}

listNode* findSpot(LLlist LL, listNode* newNode)
{
    listNode* spot = LL.listHead;
    while (spot->next != NULL && spot->next->data < newNode->data)
    {
        spot = spot->next;
    }
    return spot;
}

void insertOneNode(listNode* spot, listNode* newNode)
{
    newNode->next = spot->next;
    spot->next = newNode;
}
```

Program Output

inFile: “below is input File”

```
95 588 12 16 666
495 69 52
88 192 40 555
58 327 955 349 325 361 637
307 111 222
613 777
637 255
2123
838 91
32 333 888 999
79  444
702
```

outFile1: “below is outFile1”

```
702
444
79
999
888
333
32
91
838
2123
```

```
255
637
777
613
222
111
307
637
361
325
349
955
327
58
555
40
192
88
52
69
495
666
16
12
588
95
```

outFile2: “below is outFile2”

```
702
444
79
999
888
333
32
91
838
2123
```

```
255
637
777
613
222
111
307
637
361
325
349
955
327
58
555
40
192
88
52
69
495
666
16
12
588
95
```

outFile3: "below is outFile3"

```
listHead -> (-9999, 12) -> (12, 16) -> (16, 32) -> (32, 40) -> (40, 52) -> (52, 58) -> (58, 69) -> (69, 79) ->
(79, 88) -> (88, 91) -> (91, 95) -> (95, 111) -> (111, 192) -> (192, 222) -> (222, 255) -> (255, 307) ->
(307, 325) -> (325, 327) -> (327, 333) -> (333, 349) -> (349, 361) -> (361, 444) -> (444, 495) -> (495,
555) -> (555, 588) -> (588, 613) -> (613, 637) -> (637, 637) -> (637, 666) -> (666, 702) -> (702, 777) ->
(777, 838) -> (838, 888) -> (888, 955) -> (955, 999) -> (999, 2123) -> NULL
```

deBugFile: "below is deBugFile"

```
entering buildStack!
95
588
12
16
```

```
666
495
69
52
88
192
40
555
58
327
955
349
325
361
637
307
111
222
613
777
637
255
2123
838
91
32
333
888
999
79
444
702
leaving buildStack!
entering buildQueue!
702
444
79
999
888
333
32
91
838
2123
255
637
```

```
777
613
222
111
307
637
361
325
349
955
327
58
555
40
192
88
52
69
495
666
16
12
588
95
leaving buildQueue!
entering buildList!
entering deleteQ method!
702
leaving deleteQ method!
702
entering deleteQ method!
444
leaving deleteQ method!
444
entering deleteQ method!
79
leaving deleteQ method!
79
entering deleteQ method!
999
leaving deleteQ method!
999
entering deleteQ method!
888
leaving deleteQ method!
888
```

```
entering deleteQ method!
333
leaving deleteQ method!
333
entering deleteQ method!
32
leaving deleteQ method!
32
entering deleteQ method!
91
leaving deleteQ method!
91
entering deleteQ method!
838
leaving deleteQ method!
838
entering deleteQ method!
2123
leaving deleteQ method!
2123
entering deleteQ method!
255
leaving deleteQ method!
255
entering deleteQ method!
637
leaving deleteQ method!
637
entering deleteQ method!
777
leaving deleteQ method!
777
entering deleteQ method!
613
leaving deleteQ method!
613
entering deleteQ method!
222
leaving deleteQ method!
222
entering deleteQ method!
111
leaving deleteQ method!
111
entering deleteQ method!
307
```

```
leaving deleteQ method!
307
entering deleteQ method!
637
leaving deleteQ method!
637
entering deleteQ method!
361
leaving deleteQ method!
361
entering deleteQ method!
325
leaving deleteQ method!
325
entering deleteQ method!
349
leaving deleteQ method!
349
entering deleteQ method!
955
leaving deleteQ method!
955
entering deleteQ method!
327
leaving deleteQ method!
327
entering deleteQ method!
58
leaving deleteQ method!
58
entering deleteQ method!
555
leaving deleteQ method!
555
entering deleteQ method!
40
leaving deleteQ method!
40
entering deleteQ method!
192
leaving deleteQ method!
192
entering deleteQ method!
88
leaving deleteQ method!
88
```

```
entering deleteQ method!
52
leaving deleteQ method!
52
entering deleteQ method!
69
leaving deleteQ method!
69
entering deleteQ method!
495
leaving deleteQ method!
495
entering deleteQ method!
666
leaving deleteQ method!
666
entering deleteQ method!
16
leaving deleteQ method!
16
entering deleteQ method!
12
leaving deleteQ method!
12
entering deleteQ method!
588
leaving deleteQ method!
588
entering deleteQ method!
95
leaving deleteQ method!
95
leaving buildList!
```