Chapter 7

3b.

template<class ItemType>

void LinkedStack<ItemType>::displayMethod() const

{

Node<ItemType>\* curPtr = topPtr;

while(curPtr!=nullptr){

cout<<curPtr->getItem()<<endl;

curPtr = curPtr->getNext();

}

}

3c.

template<class ItemType>

void ArrayStack<ItemType>::displayMethod() const

{

for(int i =0; i<=top; i++){

cout<<items[i]<<endl;

}

}

Ex 8

1.

int sumIntegers(ListInterface<int>\* listPtr){

int data[] = {1,2,3,4,5,6};

cout << "isEmpty: returns " << listPtr->isEmpty() << "; should be 1 (true)" << endl;

int total = 0;

for (int i = 0; i < 6; i++)

{

if (listPtr->insert(i + 1, data[i]))

{

cout << "Inserted " << listPtr->getEntry(i + 1) << " at position " << (i + 1) << endl;

}

else

{

cout << "Cannot insert " << data[i] << " at position " << (i + 1) << endl;

}

}

int j = listPtr->getLength();

for (int i=0; i<j; i++){

total +=listPtr->getEntry(i+1);

}

return total;

}

int main(){

ListInterface<int>\* listPtr = new LinkedList<int>();

int sum = sumIntegers(listPtr);

cout<<sum<<endl;

System(“PAUSE”);

}

6.

template<class ItemType>

int getEntry(ListInterface<ItemType>\* listPtr, ItemType find){

int data[] = {2,3,5,1,2,4};

cout << "isEmpty: returns " << listPtr->isEmpty() << "; should be 1 (true)" << endl;

int total = 0;

for (int i = 0; i < 6; i++)

{

if (listPtr->insert(i + 1, data[i]))

{

cout << "Inserted " << listPtr->getEntry(i + 1) << " at position " << (i + 1) << endl;

}

else

{

cout << "Cannot insert " << data[i] << " at position " << (i + 1) << endl;

}

}

int j = listPtr->getLength();

int listNumber=-1;

for(int i =0; i<j; i++){

ItemType entry = listPtr->getEntry(i+1);

if(entry==find){

listNumber=(i+1);

}

}

return listNumber;

}

int main()

{

ListInterface<int>\* listPtr = new LinkedList<int>();

int check = getEntry(listPtr, 5);

cout<<"Answer Should be 3: "<<check<<endl;

check = getEntry(listPtr, 9);

cout<<"Answer Should be -1: "<<check<<endl;

system("PAUSE");

return 0;

}