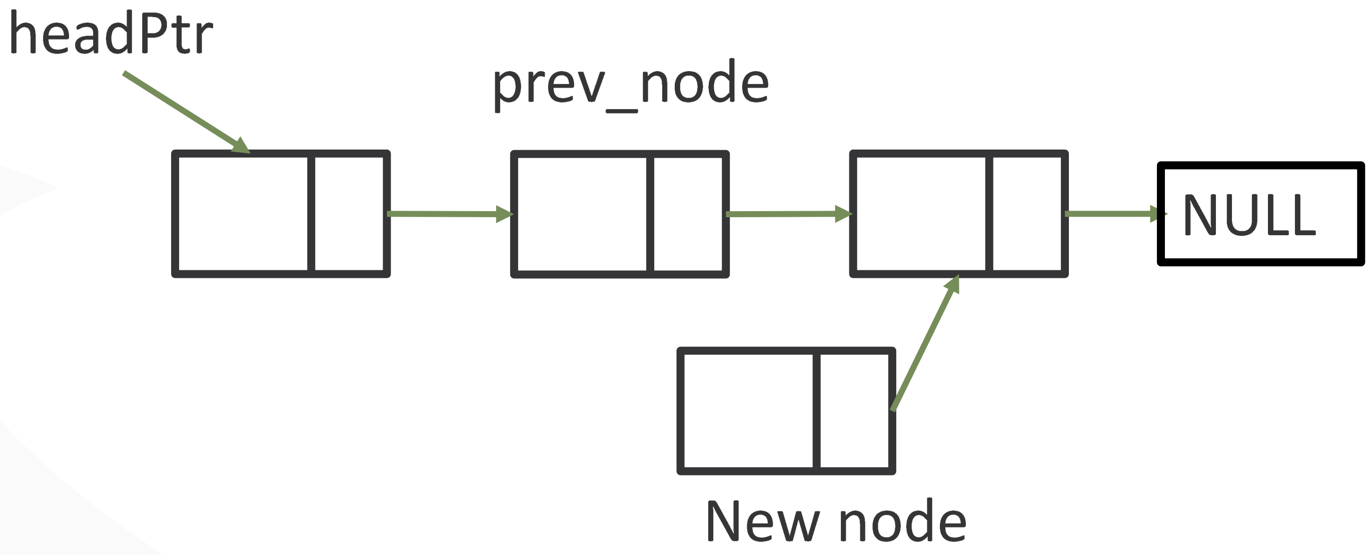
Data Structures TA Session3 – hw4

By Po-Chuan & Pei-Hsuan

1. Exercise 4.2

Write C++ implementations of the pseudocodes written in the previous exercise.

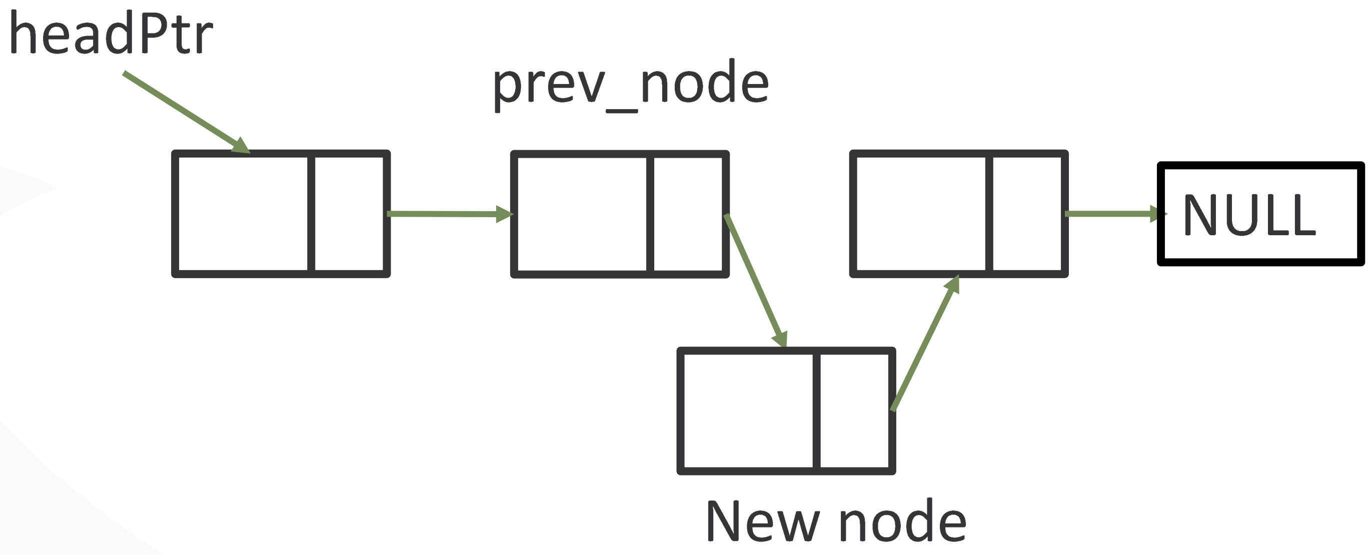
Write an algorithms for adding a node the end of a list, and inserting a node in a particular position in the list, assuming the list is ordered.



1. Exercise 4.2

Write C++ implementations of the pseudocodes written in the previous exercise.

Write an algorithms for adding a node the end of a list, and inserting a node in a particular position in the list, assuming the list is ordered.



1. Exercise 4.2

Write C++ implementations of the pseudocodes written in the previous exercise.

Write an algorithms for adding a node the end of a list, and inserting a node in a particular position in the list, assuming the list is ordered.

```cpp
template <class ItemType>
bool LinkedBag<ItemType>::add_node(const ItemType& newEntry, ItemType& prev_node)
{
      Node<ItemType>* newNodePtr = new Node<ItemType>(); // allocate new node

      newNodePtr->setItem(newEntry);   // put in the data
      newNodePtr->setNext(prev_node->getNext());
                     // make next of new node as next of prev_node
      prev_node->setNext(newNodePtr);  // move the next of prev_node as new_node

      itemCount++;
      return true;
}
```

Exercise 4.2 - Grading Policy

itemCount++	2
add a node the end of a list	7
inserting a node in a particular position in the list	7
syntax correctness	4

2. Exercise 4.3 (use C++)

Suppose that the class LinkedBag did not have the data member itemCount. Write methods:

a. To count the number of nodes

b. To display the value stored in each node in the linked chain

a.

```
template <class ItemType>
int LinkedBag<ItemType>::count_nodes()
{
	int count = 0;
	Node<ItemType>* curPtr = headPtr;
	while(curPtr != nullptr)
	{
		count++;
		curPtr = curPtr->getNext();
	}
	return count;
}
```

2. Exercise 4.3 (use C++)

Suppose that the class LinkedBag did not have the data member itemCount. Write methods:

a. To count the number of nodes

b. To display the value stored in each node in the linked chain

b.

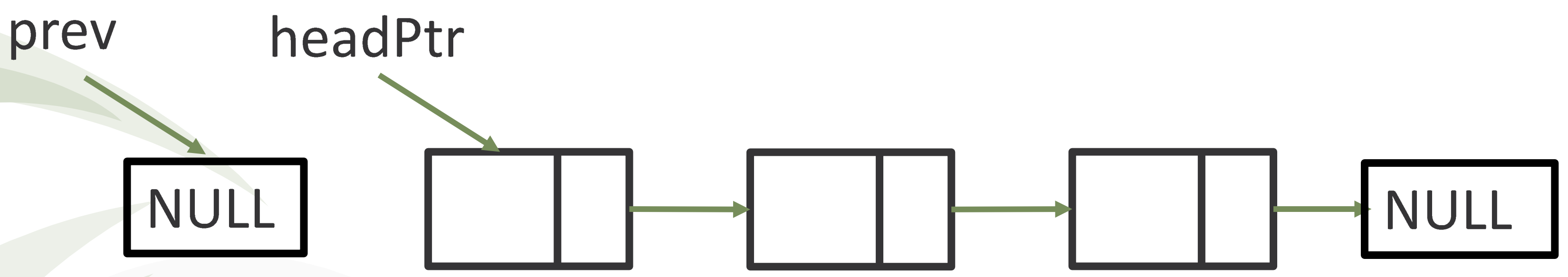
```
template <class ItemType>
int LinkedBag<ItemType>::display()
{
	Node<ItemType>* curPtr = headPtr;
	while(curPtr != nullptr)
	{
		cout << curPtr->getItem() << endl;
		curPtr = curPtr->getNext();
	}
}
```

Exercise 4.3 - Grading Policy

- each question has 10 points
- function correctness 7
- syntax correctness 3

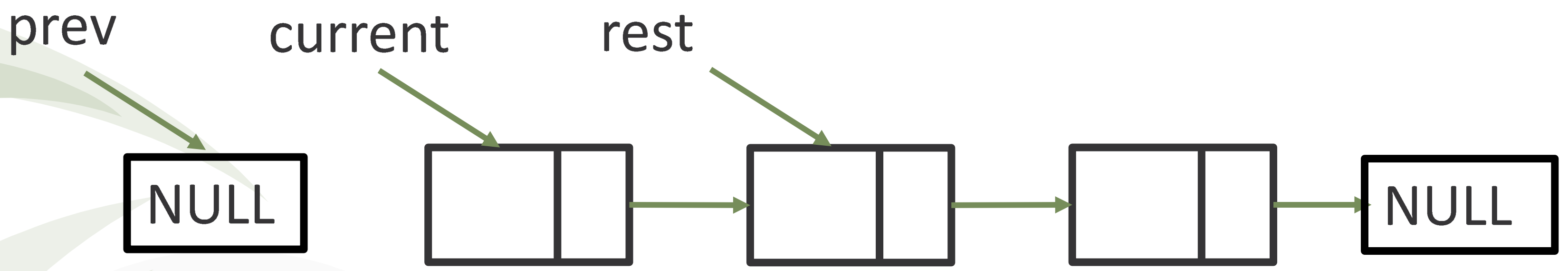
3. Exercise 4.4 (use C++)

Specify and define a method reverse that reverses the order of the nodes in a list



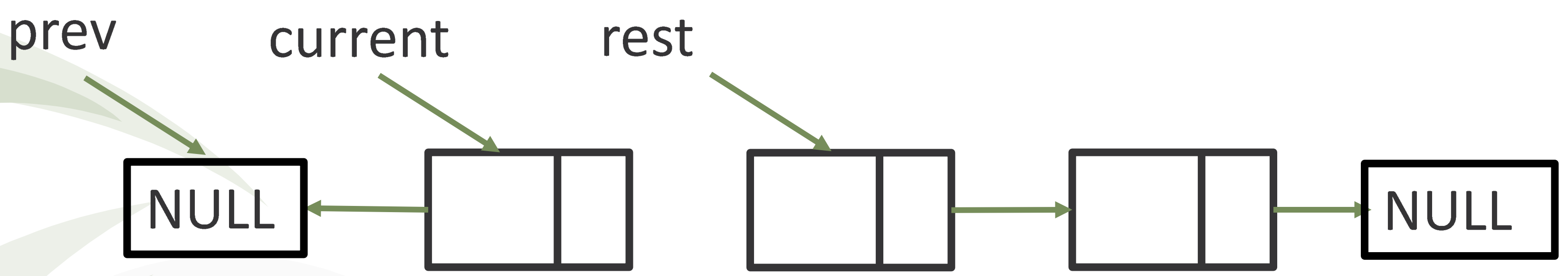
3. Exercise 4.4 (use C++)

Specify and define a method reverse that reverses the order of the nodes in a list



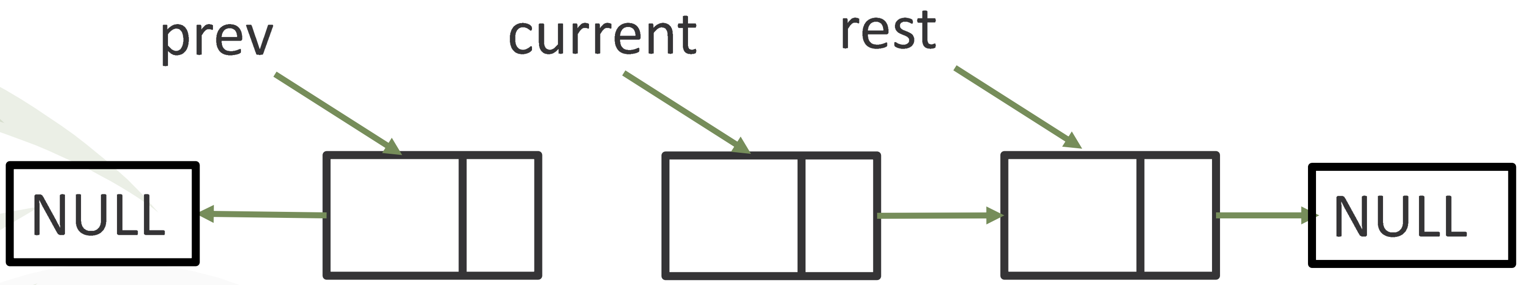
3. Exercise 4.4 (use C++)

Specify and define a method reverse that reverses the order of the nodes in a list



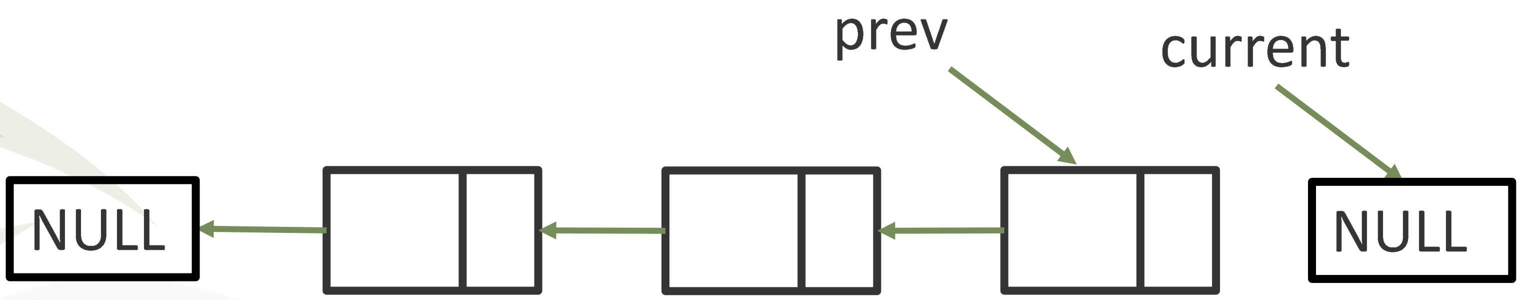
3. Exercise 4.4 (use C++)

Specify and define a method reverse that reverses the order of the nodes in a list



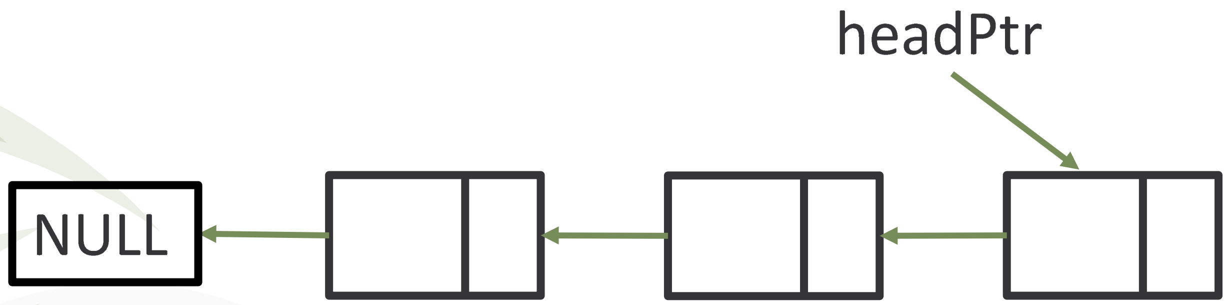
3. Exercise 4.4 (use C++)

Specify and define a method reverse that reverses the order of the nodes in a list



3. Exercise 4.4 (use C++)

Specify and define a method reverse that reverses the order of the nodes in a list



3. Exercise 4.4 (use C++)

```
template <class ItemType>
void LinkedBag<ItemType>::reverse()
{
      Node<ItemType>* prev = NULL;
      Node<ItemType>* current = headPtr;
      Node<ItemType>* rest =NULL;

      While (current != NULL)
      {
            rest = current->getNext(); // set rest equal to the next of current
            current->setNext(prev);   // set prev is the next node of current
            prev = current;    // move current node to be next prev node
            current = rest;    // move rest node to be next current node
      }

      headPtr = prev;             // let prev node be new headPtr
}
```

Exercise 4.4 - Grading Policy

- syntax correctness 3
- function correctness 10
- use linked-list to implement 5
- final headPtr value 2

4. Exercise 4.7 (use C++)

Specify and define a method that destroy an entire list and removes the memory used by the list.

```
template <class ItemType>
void LinkedBag<ItemType>::destroy_list ()
{
	Node<ItemType>* temp = headPtr;
	while(headPtr != nullptr)
	{
		headPtr = headPtr->getNext();
		delete temp;      // destroy it
		temp = headPtr;   // move headPtr node to next temp
	}
	itemCount = 0;
}
```

Exercise 4.7 - Grading Policy

- syntax correctness 5
- function correctness 10
- use linked-list to implement 5
- itemCount=0 2

5.

State the advantages of linked list-based implementations of the ADT bag over array-based ones, and the other way around.

Compared to array, list does not have a size limit, so you can use space whenever you need it. Moreover, list does not need to predict the maximum number of items, while array needs. Therefore, List performs generally better in inserting, extracting and moving elements in any position. However, list-based requires more space to store an item and needs more time than array-based to access any item.

5 - Grading Policy

Missing each answer will lose 3 points.

Data Structures
TA Session3 – hw6

By Po-Chuan & Pei-Hsuan

1. Exercise 6.4

Train stack

Identify three stacks in the figure.

- 圖中所示的左、中、右，皆為堆疊(棧)，總計三個。

How they relate to each other?

- 三個堆疊的元素數量和為定值
- 其中一個棧呼叫 pop() 的時候，必有一相鄰棧會呼叫 push() 函式，其被取出之元素將會插入該相鄰棧中；反之亦然

How can you use this system to construct any possible permutation of railroad cars?

I: initial permutation

M: tmp stack

T: target permutation

```
while I is not empty
        pop from I to M and search the desired car
        move that car into T
        move all cars in M back to I
```

Grading policy

• Identify 3 stacks	7
• How they are related	7
• Construct any possible sequence	6

2. Exercise 6.6

String Correction

6-6 a. stack contents (stack from bottom to top)

Input character	Stack contents
a	a
b	a b
c	a b c
←	a b
d	a b d
e	a b d e

Input character	Stack contents
←	a b d
←	a b
f	a b f
g	a b f g
←	a b f
h	a b f h

6-6 c. C++ implementation

```
string correct( const string& input )
{
	stack<char> s;
	string ret;

	for ( int i = 0; i < input.size(); ++i )
		if ( input[ i ] != '←' )		// letters
			s.push( input[ i ] );	// push to s
		else
			s.pop();			// delete a letter

	for( ; !s.empty(); s.pop() )
		ret = s.top() + ret;		// append to ret
	return ret;
}
```

Grading policy

• 6-6 a.	10
• String correction	5
• Reverse order	2
• Internal stack	2
• Return the string	1

3. Exercise 6.9

Palindrome

6-9 b. (stack from bottom to top)

Character	Stack contents
c	c
b	b c
b	b b c
$	b b c
b	b c ∅
b	c
c	

6-9 d. (stack from bottom to top)

Character	Stack contents
x	x
y	y x
y	y y x
z	z y y x
$	z y y x
z ∅	y y x
y	y x

Grading policy

- 2 problems in this set (10 points * 2)
- 1 point deduction for each error

4. Exercise 6.12

Infix to postfix

6-12 b. (stack from bottom to top)

∅

Ch	Stack	Postfix
(	(	
a	(	a
+	(+ ∅	a
b	(+	ab
)		ab+
*	*	ab+

Ch	Stack	Postfix
(	*(	ab+
c	*(	ab+c
-	*(- ∅	ab+c
d	*(-	ab+cd
)		ab+cd-*

6-12 c. (stack from bottom to top)

Ch	Stack	Postfix
(	(	?
a	(	a
*	(*	a
(	(*(	a
b	(*(	ab
*	(*(*	ab
c	(*(*	abc
)	(*	abc*
)		abc**
-	-	abc**
d	-	abc**d
+	+	abc**d
e	+	abc**d-e
/	+/	abc**d-e
f	+/	abc**d-ef
		abc**d-ef/+

Grading policy

- 2 problems in this set (10 points * 2)
- 1 point deduction for each error

5. Exercise 6.12

Stack axioms

Prove that any stack is equal to a stack that is in canonical form.

1. When a set is empty (length = 0), the assertion is correct.
2. Suppose a set of length of *n*, S, is in canonical form.
3. S.push() creates a set of length of *n* + 1, which is also in canonical form.
4. By M.I., we prove that any stack is equal to a stack that is in canonical form.

Simplify expression

- (aStack.push(item)).pop() = aStack:
  ((((((((((new Stack()).push(6)).**push(9)).pop()).**pop()).
  push(2)).pop()).push(3)).push(1)).pop()).peek()
  =((((((((new Stack()).**push(6)).pop()).**push(2)).pop()).push(3)).push(1)).pop()).peek()
  =((((((new Stack()).**push(2)).pop()).**push(3)).push(1)).pop()).peek()
  =((((new Stack()).push(3)).**push(1)).pop()).**peek()
  =((new Stack()).push(3)).peek()

Simplify expression

- (aStack.push(item)).peek()=item: ((new Stack()).push(3)).peek()=3
- Problem assertion is correct.

Grading policy

• Proof	10
Explanation	8
• Expression simplification	10

The end~

Hope you did a good job in this assignment.

Average score is 79.6

By the TAs

104/11/30