## 10.1-2

要是想在A[1..n]实现两个栈，则需设置两个栈指针S1.top,S2.top，分别指向A[1]与A[n]处。两个栈push元素，则分别执行，S1.top++,S2.top--，这样，当两个栈的元素个数的和不为n时，则两个栈不会发生栈上溢。

## 10.1-4

|  |
| --- |
| ENQUEUE(Q, x){  If (Q.head==Q.tail+1 or Q.head==1 or Q.tail==Q.length)  error “Queue overflow”  Q[Q.tail]=x  If (Q.tail==Q.length)  Q.tail=1  else  Q.tail=Q.tail+1  }  DEQUEUE(Q){  If (Q.tail==Q.head)  error “Queue underflow”  x=Q[Q.head]  if (Q.head=Q.length)  Q.head=1  else  Q.head=Q.head+1  return x  } |

## 10.1-6

用两个栈实现一个队列。分为入队与出队两个操作，

入队时，将元素压入栈S1中。

出队时，检查栈S2是否为空，若不为空，则弹出S2中的首元素，否则将S1中元素全部压入S2中，再弹出首元素。

## 10.1-7

用两个队列实现一个栈的原理与上题同理。

## 10.2-1

单链表Insert可能在O(1)的时间内完成，如尾插法与头插法。

## 10.2-2

设置一个cur指针与pre指针，分别指向当前元素p与其前驱。

进栈时，p->next = cur->next

cur->next = p

pre = cur

出栈时，pre->next = cur->next

return cur

## 10.2-3

设头指针head，尾指针tail。

进队，p->next = tail->next

tail->next = p

tial = p

出队, p = head->next

head->next = p->next

return p

## 10.2-5

|  |
| --- |
| INSERT(L, x){  x->next = L->next  L->next = x  }  DELETE(L, x){  pre = FINDPRE(x)  pre->next = x->next  delete x  }  SEARCH(L, x){  k = L.head  while k.next != NIL and k.key != x  k = k.next  return k  } |

## 10.2-6

采用链表。

## 10.2-7

链表的逆置。

方法一：用链表的头插法。

|  |
| --- |
| LinkList Reverse(LinkList &L)  {  LinkList r;  p = L->next;  L->next = NULL;  while(p != NULL){  r = p->next;  p->next = L->next;  L->next = p;  p = r;  }  return L;  } |

方法二：采用三个指针。

|  |
| --- |
| LinkList Reverse2(LinkList &L)  {  LNode \*pre , \*p = L->next; \*r = p->next;  p->next = NULL;  while(r != NULL){  pre = p;  p = r;  r = r->next;  p->next = pre;  }  L->next = p;  return L;  } |

## 10.4-2

|  |
| --- |
| PreOrderRecursive(T){  If T != NIL  Visit(T)  PreOrderRecursive(T->lchild)  PreOrderRecursive(T->rchild)  } |

## 10.4-3

|  |
| --- |
| PreOrderNoRecur(T)  {  Stack s;  s.push(T);  while T and !s.empty()  while (T)  visit(T)  T = T->lchild  s.push(T)  p = s.pop()  T = p->rchild  } |