第二章作业

2022211363 谢牧航

选择题

1. A
2. B
3. B
4. D

简答题

1. L -> [6] -> [5] -> [8] -> [7] -> NULL
2. 已知P结点是某双向链表的中间结点，写出一下操作的语句序列

   a. 在P结点后插入S结点的语句序列；

   ```
   p -> next = s;
   s -> prev = p;
   ```

   b. 在P结点前插入S结点的语句序列；

   ```
   p -> prev = s;
   s -> next = p;
   ```

   c. 删除P结点的直接后继结点的语句序列；

   ```
   p -> next ->next -> prev = p;
   p -> next = p -> next -> next;
   ```

   d. 删除P结点的直接前驱结点的语句序列；

```
p -> prev -> prev -> next = p;
p -> prev = p -> prev -> prev;
```

e. 删除P结点的语句序列。

```
p -> prev -> next = p -> next;
p -> next -> prev = p -> prev;
```

3. 简述以下算法的功能
   a. 把链表的第一个结点移到链表的最后一个结点之后；
   b. 把循环链表拆成两个循环链表。

算法设计题

1. 算法设计题，从一个给定的顺序表L中删除值在 x~y （x<=y）之间的所有元素，要求以较高的效率来实现，写出算法伪代码并分析你的算法时间复杂度。（提示：移动位置一步到位）

```
Procedure DeleteRange(List: array of Integer, x, y: Integer)
  index := 1
  tempLength := Length(List)
  for i := 1 to Length(List) do
    if List[i] >= x and List[i] <= y then
      tempLength := tempLength - 1
    else
      List[index] := List[i]
      index := index + 1
    end if
  end for
  Length(List) := tempLength
End Procedure
```

时间复杂度为 $O(n)$。

2. 算法设计题，设计一个空间复杂度为0(1)的算法shift(SqList L, int k)将顺序表L中的元素整体循环左移k位，要求以较高的效率来实现，写出算法伪代码，并分析在如下示例的10个数据，循环左移4位时这10个数据总共移动的次数。

```
Procedure Reverse(A: array of Integer, start, stop: Integer)
    i := start
    j := stop
    while i < j do
        temp := A[i]
        A[i] := A[j]
        A[j] := temp
        i := i + 1
        j := j - 1
    end while
End Procedure

Procedure Shift(L: array of Integer, k: Integer)
    n := Length(L)
    k := k mod n  { 确保 k 不大于 n }
    { 反转整个数组 }
    Reverse(L, 0, n - 1)
    { 反转前 n - k 个元素 }
    Reverse(L, 0, n - k - 1)
    { 反转后 k 个元素 }
    Reverse(L, n - k, n - 1)
End Procedure
```

分析在示例的10个数据中，循环左移4位时这10个数据总共移动的次数：

第一次反转整个数组，需要移动 10 次。
第二次反转前 6 个元素（10 - 4 = 6），需要移动 6 / 2 = 3 次。
第三次反转后 4 个元素，需要移动 4 / 2 = 2 次。

总共需要移动 10 + 3 + 2 = 15 次。

3. 算法设计题：设计一个算法删除一个单链表倒数第k个结点

```
Procedure DeleteKthFromEnd(var head: Pointer to Node k: Integer)
  first := head
  second := head
  prev := Nil
  { 第一个指针先向前移动 k 步 }
  for i := 1 to k do
    if first = Nil then
      WriteLn('The list has fewer than k elements.')
      Exit
    end if
    first := first -> next
  end for
  { 两个指针一同向前移动 }
  while first != Nil do
    first := first -> next
    prev := second
    second := second -> next
  end while
  { 删除目标结点 }
  if prev = Nil then
    { 如果删除的是头结点 }
    head := head -> next
  else
    prev -> next := second -> next
  end if
  Dispose(second)
End Procedure
```

4. 算法设计题，设计一个算法检测一个单链表L上是否因为某种错误操作而出现了环。

使用 Floyd 判环法：

```
Function HasCycle(head: Pointer to Node): Boolean
  if head = Nil then
    Result := False
    Exit
  end if
  slow := head
  fast := head
  while (fast != Nil) and (fast -> next != Nil) do
    slow := slow -> next
    fast := fast -> next -> next
    if slow = fast then
      Result := True
      Exit
    end if
  end while
  Result := False
End Function
```