```
polyPointer getNode(void)
{ /* 사용할 노드를 제공 */
   polyPointer node;
   if (avail) {
      node = avail;
      avail = avail→link;
   }
   else
      MALLOC(node, sizeof(*node));
   return node;
}
```

프로그램 4.12: 함수 *getNode*

```
void retNode(polyPointer node)
{ /* 가용 리스트에 노드를 반환 */
   node→link = avail;
   avail = node;
}
```

프로그램 4.13: 함수 *retNode*

```
void cerase(polyPointer *ptr)
{ /* ptr가 가리키는 원형 리스트를 제거 */
   polyPointer temp;
   if (*ptr) {
      temp = (*ptr)→link;
      (*ptr)→link = avail;
      avail = temp;
      *ptr = NULL;
   }
}
```

프로그램 4.14: 원형 리스트의 제거

```
polyPointer cpadd(polyPointer a, polyPointer b)
{ /* 다항식 a와 b는 헤더 노드를 가진 단순 연결 원형 리스트이고, a와 b가 합산된
     다항식을 반환한다. */
   polyPointer startA, c, lastC;
   int sum, done = FALSE;
   startA = a;              /* record start of a */
   a = a→link;              /* skip header node for a and b*/
     b = b→link;
   c = getNode();           /* get a header node for sum */
   c→expon = -1; lastC = c;
   do {
      switch (COMPARE(a→expon, b→expon)) {
         case -1: /* a-expon < b-expon * /
               attach(b→coef, b→expon, &lastC);
               b = b→link;
               break;
         case 0: /* a→expon = b→expon */
               if (startA == a) done = TRUE;
               else {
                  sum = a→coef + b→coef;
                  if (sum) attach (sum, a→expon, &lastC);
                  a = a→link; b = b→link;
               }
               break;
         case 1: /* a→expon > b→expon */
               attach(a→coef, a→expon, &lastC);
               a = a→link;
      }
   } while (!done);
   lastC→link = c;
   return c;
}
```

프로그램 4.15: 헤더 노드를 가진 원형 리스트로 표현된 두 다항식의 덧셈