Chapter 3,4

문제 1.

```
list_pointer create2() {
    /* create a linked list with two nodes */
    list_pointer first, second;
    ***
    second->data = 20;
    first->data = 10; first->link = second; return first;
}
```

문제 2.

```
void insert(list_pointer *ptr, list_pointer node) {
    /* insert a new node with data=50 into the list ptr after node */
    list_pointer temp;
    temp = (list_pointer) malloc(sizeof(list_node));
    if (IS_FULL(temp)) {
        fprintf(stderr, "The memory is full\n");
        exit(1);
    }
    temp->data = 50;
    ***
}
```

문제 3.

```
void delete(list_pointer *ptr, list_pointer trail, list_pointer node) {
    /* delete node from the list, trail is the preceding node ptr is the head of the
list */
    ***
    free(node);
}
```

문제 4.

```
list_pointer search(list_pointer ptr, int num) {
    for (; ptr; ptr = ptr->link) {
        ***
    }
    return ptr;
}
```

문제 5.

```
void merge(list_pointer x, list_pointer y, list_pointer *z) {
    list_pointer last;
    last = (list_pointer) malloc(sizeof(list_node));
    *z = last;

    while (x && y) {
        ***
    }

    if (x)
        last->link = x;
    else if (y)
        last->link = y;

    last = *z;
    *z = last->link;
    free(last);
}
```

문제 6.

```
void addq(int i, element item) {
    /* add item to the rear of queue i */
    queue_pointer temp = (queue_pointer) malloc(sizeof(queue));
    if (IS_FULL(temp)) {
        fprintf(stderr, "The memory is full\n");
        exit(1);
    }
    *****
}
```

문제 7.

```
element deleteq(int i) {
    /* delete an element from queue i */
    queue_pointer temp = front[i];
    element item;

    if (IS_EMPTY(front[i])) {
        fprintf(stderr, "The queue is empty\n");
        exit(1);
    }

    ***

    return item;
}
```

문제 8.

```
poly_pointer padd(poly_pointer a, poly_pointer b) {
    /* return a polynomial which is the sum of a and b */
    poly_pointer c, rear, temp;
    int sum;

    rear = (poly_pointer) malloc(sizeof(poly_node));
    if (IS_FULL(rear)) {
        fprintf(stderr, "The memory is full\n");
        exit(1);
    }

    c = rear;
    while (a && b) {
        switch (COMPARE(a->expon, b->expon)) {
            ***
        }
    }

    /* copy rest of list a and then list b */
    ***

    rear->link = NULL;

    /* delete extra initial node */
    temp = c;
    c = c->link;
    free(temp);

    return c;
}
```

문제 9.

```
list_pointer invert(list_pointer lead) {
    /* invert the list pointed to by lead */
    list_pointer middle, trail;
    middle = NULL;

    while (lead) {
        ***
    }

    return middle;
```

```
}
```

문제 10.

```
/* Phase 2: output the equivalence classes */
for (i = 0; i < n; i++) {
    if (out[i]) {
        printf("\nNew Class: %5d", i);
        out[i] = FALSE; /* set class to false */
        x = seq[i];
        top = NULL; /* initialize stack */

        for (;;) { /* find rest of class */
            while (x) { /* process list */
                ***
            }
            if (!top) break;
            x = seq[top->data];
            top = top->link; /* unstack */
        }
    }
}
```