NAME: Jinyi XIA
STUDENT ID: 2021212057
CLASS NUMBER: 2021211802
CONTAINER NUMBER: a4d4f6a5498a4e912a4d1a76a7fa36cf23af64b416688d222a8968a38b29a711

REPORT ON LAB 1

1 Environment

All operations are conducted in the given Docker image.

2 Procedure

1. Implement the linked list interfaces given.
2. Build the dynamic link library via `make libll` command.
3. Test code with the Python script `ll_test.py`.

3 Result

It turns out that the linked list implementation is correct as is shown in fig. 1.

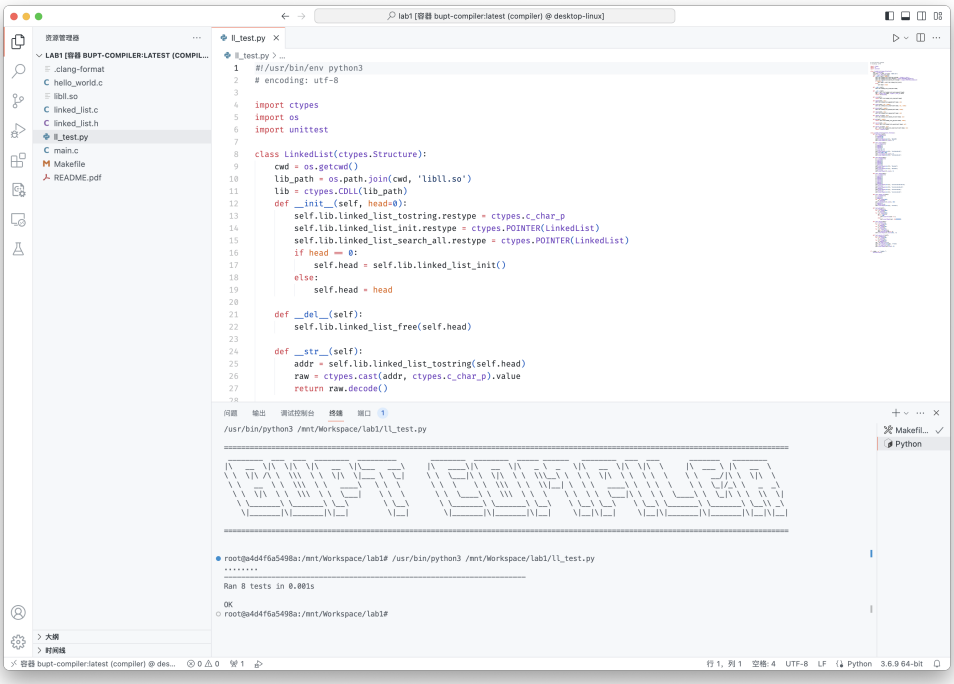

Figure 1: Test result

4 Code

```
void linked_list_insert(node *head, int val, int index) {
    if (index < 0 || index > head->count) {
        return;
    }
    node *new_node = (node *)malloc(sizeof(node)), *pos = head, *next;
    new_node->value = val;
    for (int i = 0; i < index; i++) {
        pos = pos->next;
    }
    next = pos->next;
    pos->next = new_node;
    new_node->next = next;
    head->count++;
}
```

Listing 1: Insert value at the position of index

```
void linked_list_delete(node *head, int index) {
    node *pos = head, *next;
    if (index < 0 || index >= head->count) {
        return;
    }
    for (int i = 0; i < index; i++) {
        pos = pos->next;
    }
    next = pos->next->next;
    free(pos->next);
    pos->next = next;
    head->count--;
}
```

Listing 2: Delete node at the position of index

```
void linked_list_remove(node *head, int val) {
    node *pos = head, *next;
    while (pos->next) {
        if (pos->next->value == val) {
            next = pos->next->next;
            free(pos->next);
            pos->next = next;
            head->count--;
            return;
        }
        pos = pos->next;
    }
}
```

Listing 3: Remove the first node with given value

```c
void linked_list_remove_all(node *head, int val) {
    node *pos = head, *next;
    while (pos->next) {
        if (pos->next->value == val) {
            next = pos->next->next;
            free(pos->next);
            pos->next = next;
            head->count--;
        } else {
            pos = pos->next;
        }
    }
}
```

Listing 4: Remove all nodes with given value

```c
int linked_list_get(node *head, int index) {
    if (index < 0 || index >= head->count) {
        return -0x80000000;
    }
    for (int i = 0; i < index; i++) {
        head = head->next;
    }
    return head->next->value;
}
```

Listing 5: Get value at the position of index

```c
int linked_list_search(node *head, int val) {
    int index = 0;
    while (head->next) {
        if (head->next->value == val) {
            return index;
        }
        head = head->next;
        index++;
    }
    return -1;
}
```

Listing 6: Get the index of the first node with given value

```c
node *linked_list_search_all(node *head, int val) {
    node *new_head = linked_list_init();
    node *cur = new_head;
    int index = 0;
    while (head->next) {
        if (head->next->value == val) {
            cur->next = (node *)malloc(sizeof(node));
            cur = cur->next;
            cur->value = index;
            cur->next = NULL;
```

```
        }
        head = head->next;
        index++;
    }
    return new_head;
}
```

Listing 7: Get the list of all nodes with given value