Worksheet 20 - Linked List Problem Solving

Exercise 1.
Given a **sorted doubly linked list** with duplicate nodes, implement the following functions.

- **`void dll_max_duplicate(dll_node *head);`**
  `This function counts the maximum duplicate nodes. For the following example,`
  `0<->1<->1<->2<->2<->3<->3<->3<->3->NULL`
  `, the function returns 4.`

- **`void dll_compress(dll_node **head, int max_duplicate);`**
  `This function updates the list such that there are no more than` **`max_duplicate`** `duplicate nodes. For the following example:`
  `0<->1<->1<->2<->2<->3<->3<->3<->3->NULL`
  `If max_duplicate = 4, then the list remains the same.`
  `If max_duplicate = 2, then the list is updated as:`
  `0<->1<->1<->2<->2<->3<->3->NULL`
  `Assume that` **`max_duplicate`** `is greater than or equal to 1.`

Please refer to the prelecture video (https://youtu.be/5iUWLkZEZPo) for doubly linked list.

```
typedef struct dll_node_t {
    int val;
    struct dll_node_t *next;
    struct dll_node_t **prev;
} dll_node;

void dll_insert_head(dll_node **head, int v) {
    dll_node *tmp = malloc(sizeof(*tmp));
    tmp->val = v;
    tmp->next = *head;
    if (*head)
        (*head)->prev = &(tmp->next);
    tmp->prev = head;
    *head = tmp;
}

void dll_insert_sorted(dll_node **head, int v) {
    dll_node *tmp = malloc(sizeof(*tmp));
    tmp->val = v;
```

```
    while (*head && ((*head)->val < v))
        head = &((*head)->next);
    tmp->next = *head;
    if (*head)
        (*head)->prev = &(tmp->next);
    tmp->prev = head;
    *head = tmp;
}

void dll_print(const dll_node *head) {
    while (head) {
        printf("%d ", head->val);
        if (head->next)
            assert(head->next->prev == &head->next);
        head = head->next;
    }
    printf("\n");
}

void dll_compress(dll_node **head, int max_duplicate){
    /* Your code here */

}
int dll_max_duplicate(dll_node *head){
    /* Your code here */

}

void dll_remove_all(dll_node **head){
    dll_node *tmp;
    while(*head){
        tmp = (*head)->next;
        free(*head);
        *head = tmp;
    }
}

int main() {
    dll_node *head = NULL;
    int i;
    int arr[10] = {3,1,1,2,2,2,3,3,3,0};
```

```
    for(i=0;i<10;i++)
        dll_insert_sorted(&head, arr[i]);
    dll_print(head);
    printf("Max number of duplicates: %d\n",
dll_max_duplicate(head));

    int n;
    printf("Enter a new number of duplicates: ");
    scanf("%d", &n);

    dll_compress(&head, n);
    dll_print(head);
    printf("Max number of duplicates: %d\n",
dll_max_duplicate(head));

    dll_remove_all(&head);
    dll_print(head);
    printf("Max number of duplicates: %d\n",
dll_max_duplicate(head));

}
```

```
Output:
0 1 1 2 2 2 3 3 3 3
Max number of duplicates: 4
Enter a new number of duplicates: 2
0 1 1 2 2 3 3
Max number of duplicates: 2

Max number of duplicates: 0
```