

```

//=====
//the function create an element , insert data into it and return its
//pointer
//element->next recive NULL
//return NULL if memory allocation failed
Node createElement(int data) {

    Node element = malloc( sizeof(*element) );
    if (element == NULL) return NULL;

    element->data = data;
    element->next = NULL;

    return element;
}
//-----
//copy element and returns its pointer
Node copyElement(Node element) {

    Node element_copy = malloc( sizeof(*element_copy) );
    if (element_copy == NULL) return NULL;

    element_copy->data = element->data;
    element_copy->next = element->next;

    return element_copy;
}
//-----
//release all the memory allocated for list
void listDestroy(Node list) {

    Node temp = NULL;

    while (list != NULL) {
        temp = list->next;
        free(list);
        list = temp;
    }
}
//-----
//add element to the list in original order
//return false if memory allocation failed and true otherwise
bool addToListDirect(Node* list_ptr, const Node const element) {

    Node element_copy = copyElement(element);
    if (element_copy == NULL) return false;

    if (*list_ptr == NULL) {
        *list_ptr = element_copy;
        return true;
    }

    Node list_running = *list_ptr;
    while (list_running->next != NULL) {
        list_running = list_running->next;
    }
    list_running->next = element_copy;

    return true;
}

```

```

//-----
//add element to the list in a reversed order
//return false if memory allocation failed and true otherwise
bool addToListRevers(Node* list_ptr, const Node const element) {

    Node element_copy = copyElement(element);
    if (element_copy == NULL) return false;

    element_copy->next = *list_ptr;
    *list_ptr = element_copy;

    return true;
}
//-----
//printing all the elements of a given list
void printList(const Node const list) {

    Node list_running = list;
    while (list_running != NULL) {
        printf("%d ", list_running->data);
        list_running = list_running->next;
    }
}
//-----
//build a a list from element in even places and a reversed list from
element
//placed in odd places
//prints the list and then the reversed list
bool printCheckmarkOrdered(const Node const list) {
    if (list == NULL) return true;
    Node list_running = list;
    int counter = 0;
    Node list_direct = NULL;
    Node list_revers = NULL;
    Node element = NULL;
    bool result;
    while (list_running != NULL) {
        element = createElement(list_running->data);
        if (element == NULL) return false;
        if (counter % 2 == 0) {
            result = addToListDirect(&list_direct, element);
        } else {
            result = addToListRevers(&list_revers, element);
        }
        if (result == false) {
            free(element);
            listDestroy(list_direct);
            listDestroy(list_revers);
            return false;
        }
        counter = counter + 1;
        list_running = list_running->next;
        free(element);
    }
    printList(list_direct);
    printList(list_revers);
    listDestroy(list_direct);
    listDestroy(list_revers);
    return true;
}
//=====

```