Linked list

No1:

int insertSortedLL(LinkedList \*ll, int item)

{

    /\* add your code here \*/

    ListNode \*temp;

    int curindex;

    curindex = 0;

    if(ll == NULL) return -1;

    else{

        temp = ll->head;

            while(curindex <= ll->size){

                if(curindex == ll->size){

                    insertNode(ll, curindex, item);

                    break;

                }

                else{

                    if(temp->item > item){

                        insertNode(ll, curindex, item);

                        break;

                    }

                    else if(temp->item == item){

                        curindex = -1;

                        break;

                    }

                    else{

                        curindex = curindex + 1;

                        temp = temp->next;

                    }

                }

            }

        }

    return curindex;

}

No2:

void alternateMergeLinkedList(LinkedList \*ll1, LinkedList \*ll2)

{

    /\* add your code here \*/

    ListNode \*temp;

    int fixsize1 = ll1->size;

    int fixsize2 = ll2->size;

    int inspos = 1;

    if(fixsize1 >= fixsize2){

        while(ll2->size >0){

            temp = findNode(ll2, 0);

            insertNode(ll1, inspos, temp->item);

            removeNode(ll2, 0);

            inspos = inspos + 2;

        }

    }

    else{

        while(fixsize1>0){

            temp = findNode(ll2, 0);

            insertNode(ll1, inspos, temp->item);

            removeNode(ll2, 0);

            inspos = inspos + 2;

            fixsize1 = fixsize1 - 1;

        }

    }

}

No3:

void moveOddItemsToBack(LinkedList \*ll)

{

    /\* add your code here \*/

    ListNode \*temp;

    temp = ll->head;

    int oddcount = 0;

    int i;

    int oddindex = 0;

    int odditem;

    while(temp != NULL){

        if(temp->item %2 == 1 || temp->item %2 == -1){

            oddcount ++;

        }

        temp = temp->next;

    }

    temp = ll->head;

    for(i=0;i<oddcount;i++){

        while(temp->item % 2 != 1 && temp->item %2 != -1){

            temp = temp->next;

            oddindex = oddindex + 1;

        }

        odditem = temp->item;

        removeNode(ll, oddindex);

        insertNode(ll, ll->size, odditem);

        temp = ll->head;

oddindex = 0;

    }

}

No4:

void moveEvenItemsToBack(LinkedList \*ll)

{

    /\* add your code here \*/

    ListNode \*temp;

    temp = ll->head;

    int evencount = 0;

    int i;

    int evenindex = 0;

    int evenitem;

    while(temp != NULL){

        if(temp->item %2 == 0){

            evencount ++;

        }

        temp = temp->next;

    }

    temp = ll->head;

    for(i=0;i<evencount;i++){

        while(temp->item % 2 != 0 ){

            temp = temp->next;

            evenindex = evenindex + 1;

        }

        evenitem = temp->item;

        removeNode(ll, evenindex);

        insertNode(ll, ll->size, evenitem);

        temp = ll->head;

        evenindex = 0;

    }

}

No5:

void frontBackSplitLinkedList(LinkedList \*ll, LinkedList \*resultFrontList, LinkedList \*resultBackList)

{

    /\* add your code here \*/

    ListNode \*temp;

    int fixsize = ll->size;

    int i;

    if(fixsize % 2 == 1){

        for(i = 0;i<=fixsize/2;i++){

            temp = findNode(ll, i);

            insertNode(resultFrontList, i, temp->item);

        }

        for(i = 1+fixsize/2;i<fixsize;i++){

            temp = findNode(ll, i);

            insertNode(resultBackList, i-1-fixsize/2, temp->item);

        }

    }

    else{

        for(i = 0;i<fixsize/2;i++){

            temp = findNode(ll, i);

            insertNode(resultFrontList, i, temp->item);

        }

        for(i = fixsize/2;i<fixsize;i++){

            temp = findNode(ll, i);

            insertNode(resultBackList, i-1+fixsize/2, temp->item);

        }

    }

}

No6:

ListNode \*temp;

    ListNode \*front;

    ListNode \*max;

    temp = \*ptrHead;

    max = \*ptrHead;

    front = \*ptrHead;

    if(temp->next == NULL){

    }

    else{

        int maxterm = temp->item;

        while(temp != NULL){

            if(temp->item > maxterm){

                max = temp;

                maxterm = temp->item;

            }

            temp = temp->next;

        }

        temp = \*ptrHead;

        while(front->next != max){

            front= front->next;

        }

        front->next = max->next;

        max->next = temp;

        \*ptrHead = max;

    }

    return 0;

}

No7.

void RecursiveReverse(ListNode \*\*ptrHead)

{

    ListNode \*first;

    ListNode \*rest;

    first = \*ptrHead;

    if(\*ptrHead == NULL) return;

    rest = first->next;

    if(rest == NULL) return;

    RecursiveReverse(&rest);

    first->next->next = first;

    first->next = NULL;

    \*ptrHead = rest;

}