S&Q

No1:

void createQueueFromLinkedList(LinkedList \*ll, Queue \*q)

{

    /\* add your code here \*/

    ListNode \*temp;

    temp = ll->head;

    while(temp != NULL){

        enqueue(q, temp->item);

        temp = temp->next;

    }

}

void removeOddValues(Queue \*q)

{

    /\* add your code here \*/

    int i;

    int temp;

    int fixsize = q->ll.size;

    for(i=0;i<fixsize;i++){

        temp = dequeue(q);

        if(temp %2 == 0){

            enqueue(q, temp);

        }

    }

}

No2:

void createStackFromLinkedList(LinkedList \*ll, Stack \*s)

{

    /\* add your code here \*/

    ListNode \*temp;

    temp = ll->head;

    while(temp != NULL){

        push(s, temp->item);

        temp = temp->next;

    }

}

void removeEvenValues(Stack \*s)

{

    /\* add your code here \*/

    Stack temp;

    int p,i;

    int fixsize = s->ll.size;

    int fixsizeoftemp;

    for(i=0;i<fixsize;i++){

        p = pop(s);

        if(p%2 == 1 || p%2 == -1){

            push(&temp,p);

        }

    }

    fixsizeoftemp = temp.ll.size;

    for(i = 0;i<fixsizeoftemp;i++){

        p = pop(&temp);

        push(s,p);

    }

}

No3:

int isStackPairwiseConsecutive(Stack \*s)

{

    /\* add your code here \*/

    int fixsize;

    int i;

    int p,q;

    fixsize = s->ll.size;

    if(fixsize % 2 == 1){

        return 0;

    }

    else{

        for(i=0;i<fixsize/2;i++){

            p = pop(s);

            q = pop(s);

            if(abs(p-q)!=1) return 0;

        }

    }

    return 1;

}

No4:

void reverse(Queue \*q)

{

    /\* add your code here \*/

    Stack s;

    s.ll.head = NULL;

    s.ll.tail = NULL;

    s.ll.size = 0;

    while(!isEmptyQueue(q)){

        push(&s, dequeue(q));

    }

    while(!isEmptyStack(&s)){

        enqueue(q, pop(&s));

    }

}

No5:

void recursiveReverse(Queue \*q)

{

    /\* add your code here \*/

    int temp;

    if(q->ll.size == 0) return;

    temp = dequeue(q);

    recursiveReverse(q);

    enqueue(q, temp);

}

No6.

void removeUntil(Stack \*s, int value)

{

    /\* add your code here \*/

    int temp;

    while(s->ll.size != 0){

        temp = pop(s);

        if(temp == value){

            push(s, temp);

            break;

        }

    }

}

No7.

int balanced(char \*expression)

{

    /\* add your code here \*/

    Stack s;

    char temp;

    int size=0;

    int i;

    while(expression[size] != '\0'){

        size = size + 1;

    }

    for(i = 0;i<size;i++){

        temp = expression[i];

        if(temp == '{'||temp == '['||temp == '('){

            push(&s, temp);

        }

        else{

            if((temp == '}'&&peek(&s)=='{')||(temp == ']'&&peek(&s)=='[')||(temp == ')'&&peek(&s)=='(')){

                pop(&s);

            }

        }

    }

    if(s.ll.size == 0) return 0;

    else return 1;

}