19/2/24

Binary Search Tree

```
struct node {
    int data;
    struct node * left;
    struct node * right;
};

struct node * createNode (int data) {
    struct node * newnode = (struct node *)
                malloc (sizeof (struct Node));
    newNode -> data = data;
    newnode -> left = NULL;
    newnode -> right = NULL;  return newnode;
}

struct node * insert (struct node * root, int data) {
    if (root == NULL) {
        return createnode (data);
    }
    if (data < root -> data) {
        root -> left = insert (root -> left, data);
    }
    else {
        root -> right = insert (root -> right, data);
    }
    return root;
}
```

```
void inOrder ( struct node * root ) {
    if (root != NULL) {
        inOrder ( root -> left ) ;
        printf ( " %d ", root -> data ) ;
        inOrder ( root -> right ) ;
    }
    printf ( " %p \n " ) ;
}

void postOrder ( struct node * root ) {
    if ( root != NULL ) {
        postOrder ( root -> left ) ;
        postOrder ( root -> right ) ;
        printf ( " %d ", root -> data ) ;
    }
}

void preOrder ( struct node * root ) {
    if (root != NULL) { printf(" %d ", root->data)
        preOrder ( root -> left ) ;
        preOrder ( root -> right ) ;
    }
}
int main ( ) {
    struct node * root = NULL ;
    root = insert ( root, 30)
    root = insert ( root, 40) ;
    root = insert ( root, 20) ;
    root = insert ( root, 30) ;
```

Odd Even Linked List

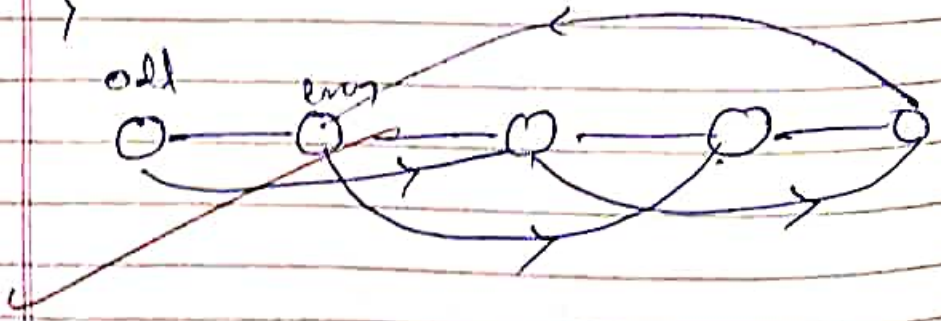
```
struct node * oddEvenList (struct node * head){
    if (head == NULL || head -> next == NULL ||
        head -> next -> next == NULL) {
            return head;
    }

    struct node *odd = head;
    struct node *even = head -> next;
    struct node * oddHead = odd;
    struct node * evenHead = even;

    while (even != NULL && even -> next != NULL){

        odd -> next = even -> next;
        odd = odd -> next;

        even -> next = odd -> next;
        even = even -> next;
    }
    odd -> next = evenHead;
    ;
    return oddHead;
}
```



19.02.24