Leet code

2, 7, 11, 15 = 23

Date
Page

1) two sum-

2 + 7 = 9
2 + 11 = 13

```
int* twoSum(int* nums, int numsSize, int target, int* returnSize){
    static int result[2];

    for(int b=0; b<numsSize; b++){
        for(int c=b+1; c<numsSize; c++){
            if(nums[b]+nums[c]==target){
                result[0]=b;
                result[1]=c;
                *returnSize=2;
                return result;
            }
        }
    }

    *returnSize=0;
    return NULL;
}
```

2) Longest Common Prefix

[flower, flow, flight]

```
#include <stdio.h>
#include <string.h>

char* longestCommonPrefix(char** strs, int strsSize){
    if(strsSize==0) return "";

    static char prefix[201];
    strcpy(prefix, strs[0]);
```

```
for(int i=1; i< strsSize; i++){
    int j=0;
    while(prefix[j]!='\0' && prefix[j]==
        strs[i][j]){
        j++;
    }
    prefix[j]='\0';
    if(prefix[0]=='\0')return "";
}
return prefix;
}
```

Leet code - 2

1) Middle of the linked list

```
struct ListNode* middleNode(struct ListNode* head){
    int count = 0
    struct ListNode* temp = head;

    while(temp != NULL){
        count++;
        temp = temp->next;
    }

    int middle = count/2

    temp = head
    for(int i = 0 ; i<middle ; i++){
        temp = temp->next;
    }
    return temp;
}
```

2) Remove Linked List Elements

```
struct ListNode* removeElements(struct ListNode*
        head, int val){

    struct ListNode dummy;
    dummy.next = head;

    struct ListNode* current = &dummy

    while(current->next != NULL){
```

Date
Page

```
if( current -> next -> val == val){

    struct ListNode* temp = current -> next;
    current -> next = temp -> next;
    free (temp);
  }else{
    current = current -> next;
  }
}

return dummy.next;
```

LEET CODE - 3

1) Linked list Cycle

```
bool hasCycle(struct ListNode *head){
    struct ListNode *visited[10000];
    int count=0;

    struct ListNode *cur=head

    while(cur!=NULL){
      for(int i=0; i<count; i++){
        if(visited[i]==cur){
          return true;
    }

    visited[count++]=cur;
    cur=cur->next;
  }
  return false;
}
```

LEET CODE - 4

1) Merge Two Binary Tree

```
struct TreeNode* mergeTrees(struct TreeNode* root1,
        struct TreeNode* root2){
    if(!root1) return root2;
    if(!root2) return root1;
    struct TreeNode* merged = (struct TreeNode*)
        malloc(sizeof(struct TreeNode));
    merged->left = mergeTrees(root1->left, root2->left);
    merged->right = mergeTrees(root1->right, root2->right);

    return merged;
}
```

Both o/p ker