6. Remove Nth Node From End of List  
Given a linked list, remove the nth node from the end and return its head.

#include<bits/stdc++.h>

using namespace std;

class Listnode{

    public:

    int data;

    Listnode\*next;

    Listnode(int val)

    {

        data=val;

        next=NULL;

    }

};

Listnode\* removenode(Listnode\* head,int N)

{

    Listnode\* slow=head;

    Listnode\* fast=head;

    //considering n cannot be greater than the size of the linkedlist

    //else we need to take in considertion the case where n is greater than the size of the linked list

    for(int i=0;i<N;i++)

    {

        fast=fast->next;

    }

    if(fast==NULL) return head->next;

    while(fast->next!=NULL)

    {

         slow=slow->next;

         fast=fast->next;

    }

    Listnode\* deletenode=slow->next;

     slow->next=slow->next->next;

     delete deletenode;

     return head;

}

int main()

{

    Listnode\* l1=new Listnode(9);

    l1->next=new Listnode(10);

    l1->next->next=new Listnode(2);

    Listnode \*start=removenode(l1,2);

    while(start!=NULL)

    {

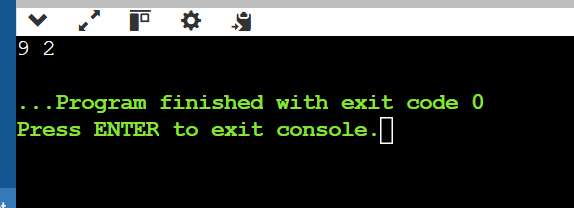
        cout<<start->data<<" ";

        start=start->next;

    }

    return 0;

}

  
  
  
  
8. Group Anagrams  
Given an array of strings, group the anagrams together.  
  
#include <bits/stdc++.h>

using namespace std;

vector<vector<string>>groupanagrams(vector<string>&s)

{

      vector<vector<string>>fans;

      if(s.empty()) return fans;

      map<string,int>mpp;

      for(int i=0;i<s.size();i++)

      {

          string ans=s[i];

          sort(ans.begin(),ans.end());

          if(mpp.find(ans)==mpp.end())

          {

             mpp[ans]=fans.size();

             fans.push\_back({});

          }

         fans[mpp[ans]].push\_back(s[i]);

      }

      return fans;

}

int main()

{

    vector<string>s={ "sanskar","eat","sleep","code","repeat","ate","tea"};

    vector<vector<string>>res=groupanagrams(s);

    for(int i=0;i<res.size();i++)

    {

         for(int j=0;j<res[i].size();j++)

         {

               cout<<res[i][j]<<" ";

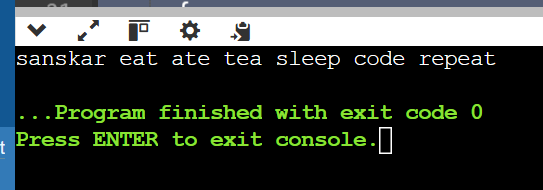
            //   cout<<endl;

         }

    }

    return 0;

}

  
  
  
4. Linked Lists  
Add Two Numbers  
You are given two non-empty linked lists representing two non-negative integers. Add the two numbers and return the sum as a linked list.  
  
#include<bits/stdc++.h>

using namespace std;

class Listnode{

    public:

    int data;

    Listnode\*next;

    Listnode(int val)

    {

        data=val;

        next=NULL;

    }

};

Listnode\* addlinkedlist(Listnode\* l1,Listnode\* l2)

{   Listnode \*dummy=new Listnode(0);

    Listnode \*temp=dummy;

    int carry=0;

    while(l1!=NULL || l2!=NULL || carry)

    {   int sum=0;

        if(l1!=NULL)

        {

             sum+=l1->data;

             l1=l1->next;

        }

         if(l2!=NULL)

        {

             sum+=l2->data;

             l2=l2->next;

        }

        sum+=carry;

        carry=sum/10;

        Listnode\*node=new Listnode(sum%10);

        temp->next=node;

        temp=temp->next;

    }

    return dummy->next;

}

int main()

{

    Listnode\* l1=new Listnode(9);

    Listnode\* l2=new Listnode(1);

    l1->next=new Listnode(10);

    l1->next->next=new Listnode(2);

    l2->next=new Listnode(12);

    l2->next->next=new Listnode(21);

    9 10 2

    1 12 21

    0 3  5 2

    //considering the linkedlist given in reverse format

    //also the resultant linkedlist is in reverse format

   Listnode\*head=addlinkedlist(l1,l2);

   while(head!=NULL)

   {

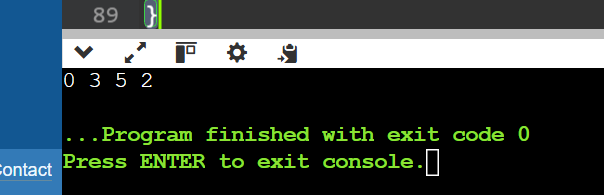
        cout<<head->data<<" ";

        head=head->next;

   }

    return 0;

}

  
  
  
9. Rearrange a no to find min possible no in o(n) and constant space.  
  
#include <bits/stdc++.h>

using namespace std;

string findmin(string num)

{

    int arr[10]={0};

    for(int i=0;i<num.size();i++)

    {

        arr[num[i]-48]++;

    }

    string ans;

    for(int i=0;i<10;i++)

    {

         for(int j=0;j<arr[i];j++)

         {

             ans+=to\_string(i);

         }

    }

    return ans;

}

int main()

{

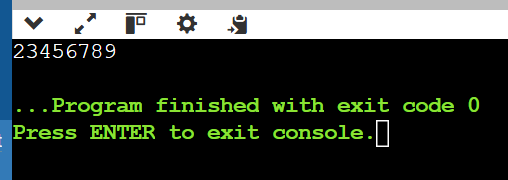
    string s="92345678";

     string fans=findmin(s);

     cout<<fans;

    return 0;

}

  
  
  
5. Reorder List  
Reorder a linked list from L0 → L1 → … → Ln-1 → Ln to L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → ….

#include <bits/stdc++.h>

using namespace std;

 class ListNode{

     public:

       int data;

       ListNode\* next;

       ListNode(int val)

       {

        data=val;

        next=NULL;

       }

 };

    void reorderList(ListNode\* head) {

        if (!head || !head->next || !head->next->next)

            return;

        ListNode\* slow = head;

        ListNode\* fast = head;

        while (fast && fast->next) {

            slow = slow->next;

            fast = fast->next->next;

        }

        ListNode\* prev = nullptr;

        ListNode\* ptr = slow->next;

        while (ptr) {

            ListNode\* next = ptr->next;

            ptr->next = prev;

            prev = ptr;

            ptr = next;

        }

        slow->next = nullptr;

        ListNode\* first = head;

        ListNode\* second = prev;

        while (second) {

            ListNode\* tmp1 = first->next;

            ListNode\* tmp2 = second->next;

            first->next = second;

            second->next = tmp1;

            first = tmp1;

            second = tmp2;

        }

    }

int main()

{

   ListNode\* l1=new ListNode(9);

    l1->next=new ListNode(10);

    l1->next->next=new ListNode(2);

    l1->next->next->next=new ListNode(13);

    reorderList(l1);

    while(l1!=NULL)

    {

        cout<<l1->data<<" ";

        l1=l1->next;

    }

}

