IT DS

Aditya Singh
2K19/EP/005

Q4

```
#include <bits/stdc++.h>
using namespace std;

class Node {
    public:
        int data; Node* next;
    Node (int x)
    {
        data = x; next = NULL;
    }

Node* push ( Node* head, int x){
    Node * curr = new Node (x);
    curr -> next = head;
    head = curr;    return head;

Node* addLists ( Node* l1, Node* l2 )
    if ( l1 == NULL) return l2;
    if (l2 == NULL) return l1;
    Stack <int> s1, s2, s3;
    Node * temp = l1;
        while ( temp != NULL)
            s1.push (temp -> data); temp = temp -> next;
    temp = l2;
        while ( temp != NULL )
            s2.push ( temp -> data);
            temp = temp -> next;
    int sum=0; carry= 0, val1, val2;
```

```
while ( !s1.empty() && !s2.empty() )
        val1 = s1.pop();   val2 = s2.pop();
        sum = (val1 + val2 + carry) % 10;
        carry = (val1 + val2 + carry) / 10;
        s3.push(sum);

while (!s1.empty())
        val1 = s1.pop();
        sum = (val1 + carry) % 10;
        carry = (val1 + carry) / 10;
        s3.push(sum);

while (!s2.empty())
        val2 = s2.pop();
        sum = (val2 + carry) % 10;  carry = (val2 + carry) / 10;
        s3.push(sum);

if(carry > 0)  s3.push(carry);


int main() {
    Node * head = new Node();  head2 = NULL;
    int first[] = {7, 9, 9, 1};  int n = sizeof(first)
    int second[] = {9, 8, 1};    int m = sizeof(second);

    for( i = m; i>=0, i--)
        push(&head1, first[i]);
    for( i = n-1; i>=0, i--)
        push(&head2, second[i]);
    printList(addList(head1, head2));
```

#include <bits/stdc++.h> using namespace std;

Q2

```
long power ( long a, long b )
    if ( b==0)  return 1;
    long res = power ( a, b/2 );
    if ( b%2 )  return res * res * a;
    else        return res * res;

int main() {
    long x = 3;
    long y = 5;
    cout << power (x, y);
```

Q3

```
int partition ( int a[], int low, int high )
    int pivot, index, i;
    index = low, pivot = high;
    for( i=low; i<high; i++ )
        if ( a[i] < a[pivot] )
            swap ( a[i], a[index] );
            index++;
    swap ( a[pivot], a[index] );
    return index;

int Quicksort ( int a[], int low, int high )
    int pindex;
    if ( low < high )
        pindex = Partition ( a, low, high );
        Quicksort ( a, low, pindex - 1 );
        Quicksort ( a, pindex+1, high );
    return 0;
```

Algorithm :

1) pick first element as pivot.

2) partition array.

```
if (low < high) {
    partition index, calculate.
    quicksort ( arr, low, pi-1);
    quicksort ( arr, pi+1, high);
}
```

It is divide & conquer algorithm. it picks an element as pivot and partitions the given array around the picked pivot.