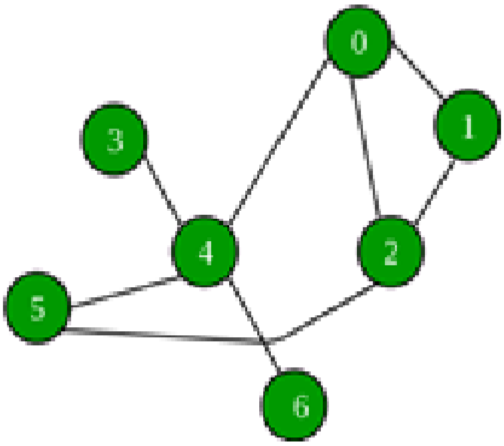
1.

You are given with an array arr which contains integer elements. Sort these elements in ascending order using insertion sort and print the 6th Iteration result.
Example:
Input:98,23,45,14,6,67,33,42
Output:6,14,23,33,45,67,98,42

```
#include <stdio.h>
void insertionSort(int arr[], int n) {
   int i, key, j;
   for ( i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;
      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
      if (i == 5) {
         printf("6th Iteration: ");
         for (int k = 0; k < n; k++) {
            printf("%d ", arr[k]);
         }
         printf("\n");
      }
   }
}
int main() {
   int arr[] = {98, 23, 45, 14, 6, 67, 33, 42};
   int n = sizeof(arr) / sizeof(arr[0]);
   insertionSort(arr, n);
   printf("Sorted array: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
   return 0;
}
```

2.
You are given an undirected graph G(V, E) with N vertices and M edges. We need to find the minimum number of edges between a given pair of vertices (u, v).
Examples:

Input: For given graph G. Find minimum number of edges between (1, 5).



```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX_VERTICES 1000

typedef struct {
    int vertex;
    struct AdjListNode *next;
} AdjListNode;

typedef struct {
    AdjListNode *head;
} AdjList;

AdjList graph[MAX_VERTICES];
int level[MAX_VERTICES];
bool visited[MAX_VERTICES];

void addEdge(int src, int dest) {
    AdjListNode *newNode = (AdjListNode *) malloc(sizeof(AdjListNode));
    newNode->vertex = dest;
    newNode->next = graph[src].head;
    graph[src].head = newNode;

    newNode = (AdjListNode *) malloc(sizeof(AdjListNode));
    newNode->vertex = src;
    newNode->next = graph[dest].head;
    graph[dest].head = newNode;
}
```

```
int min_edges(int u, int v) {
    int i;
    AdjListNode *temp;

    for (i = 0; i < MAX_VERTICES; i++) {
        graph[i].head = NULL;
        visited[i] = false;
        level[i] = -1;
    }

    level[u] = 0;
    visited[u] = true;

    // Create a queue and enqueue the source vertex
    temp = graph[u].head;
    // Enqueue the source vertex
    printf("%d ", u);

    while (temp != NULL) {
        if (temp->vertex == v)
            return level[v];

        if (!visited[temp->vertex]) {
            visited[temp->vertex] = true;
            level[temp->vertex] = level[u] + 1;
            printf("%d ", temp->vertex);

            // Dequeue a vertex from the queue and enqueue its adjacent vertices
            temp = temp->next;
        }
    }

    return -1;
}

int main() {
    int n, m, u, v, i, j;

    printf("Enter the number of vertices and edges: ");
    scanf("%d %d", &n, &m);

    printf("Enter the edges (u, v):\n");
    for (i = 0; i < m; i++) {
        scanf("%d %d", &u, &v);
        addEdge(u, v);
    }
    printf("Enter the source and destination vertices: ");
    scanf("%d %d", &u, &v);

    printf("Minimum number of edges between %d and %d is: %d\n", u, v, min_edges(u, v));

    return 0;
}
```

3.
Given the head of a singly linked list, return number of nodes present in a linked
Example 1:
1->2->3->5->8
Output 5

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int val;
    struct Node* next;
} ListNode;
int count_nodes(ListNode* head) {
    int count = 0;
    ListNode* current = head;
    while (current != NULL) {
        count++;
        current = current->next;
    }
    return count;
}
int main() {
    ListNode* head = (ListNode*)malloc(sizeof(ListNode));
    head->val = 1;
    head->next = (ListNode*)malloc(sizeof(ListNode));
    head->next->val = 2;
    head->next->next = (ListNode*)malloc(sizeof(ListNode));
    head->next->next->val = 3;
    head->next->next->next = (ListNode*)malloc(sizeof(ListNode));
    head->next->next->next->val = 5;
    head->next->next->next->next = (ListNode*)malloc(sizeof(ListNode));
    head->next->next->next->next->val = 8;
    head->next->next->next->next->next = NULL;
    printf("Number of nodes: %d\n", count_nodes(head));
    return 0;
}
```

4.

Given a number n. the task is to print the Fibonacci series and the sum of the series using recursion.

input: n=10
output: Fibonacci series
0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Sum: 88

```
#include <stdio.h>
int fibonacci(int n);
int fibonacciSum(int n);
int main() {
    int n = 10;;
    for (int i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
   printf("\n\nSum: %d\n", fibonacciSum(n));
    return 0;
}
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}
int fibonacciSum(int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += fibonacci(i);
    }
return sum;
}
```

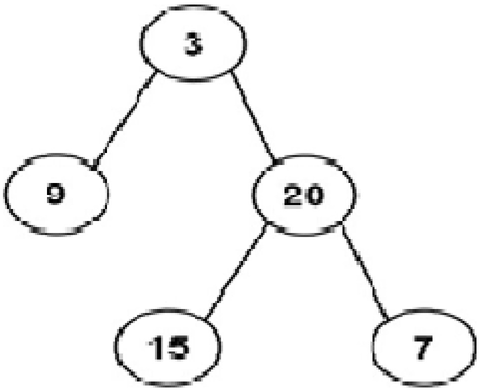
5.

You are given an array arr in increasing order. Find the element x from arr using binary search.

Example 1: arr={ 1,5,6,7,9,10},X=6
Output : Element found at location 2
Example 2: arr={ 1,5,6,7,9,10},X=11
Output : Element not found at location 2

```
#include <stdio.h>

int binarySearch(int arr[], int x, int n) {
  int low = 0;
  int high = n - 1;

  while (low <= high) {
    int mid = (low + high) / 2;

    if (arr[mid] == x) {
      return mid;
    } else if (arr[mid] < x) {
      low = mid + 1;
    } else {
      high = mid - 1;
    }
  }

  return -1;
}

int main() {
  int arr[] = {1, 5, 6, 7, 9, 10};
  int x = 6;
  int n = sizeof(arr) / sizeof(arr[0]);

  int result = binarySearch(arr, x, n);

  if (result != -1) {
    printf("Element found at location %d\n", result);
  } else {
    printf("Element not found\n");
  }

  return 0;
}
```

6..Write a program to traverse the nodes present in the following tree in inorder and postorder traversal



```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
        int key;
        struct Node *left;
        struct Node *right;
};
struct Node *newnode(int key)
{
        struct Node *node=(struct Node *)malloc(sizeof(struct Node));
        node->key=key;
        node->left=NULL;
        node->right=NULL;
        return node;
}
void inorder(struct Node *root)
{
        if(root==NULL)
        return;
        inorder(root->left);
        printf("%d ",root->key);
        inorder(root->right);
}
void postorder(struct Node *root)
{
        if(root==NULL)
        return;
        postorder(root->left);
        postorder(root->right);
        printf("%d ",root->key);
}
int main()
```

```
{
            struct Node *root=newnode(3);
            root->left=newnode(9);
            root->right=newnode(20);
    root->right->left=newnode(15);
    root->right->right=newnode(7);
            printf("inder traversal :\n");
            inorder(root);
            printf("\n");
            printf("postorder traversal :\n");
            postorder(root);
 return 0;
}
```

7.

Given a string s, sort it in ascending order and find the starting index of repeated character

Input: s = "tree"

Output: "eert", starting index 0

Input: s = "kkj"

Output: "jkk", starting index : 1

Example 2:

Input: s = "cccaaa"

Output: "aaaccc", starting index 0,3

Example 3:

Input: s = "Aabb"

Output: "bbAa",starting index 0,2

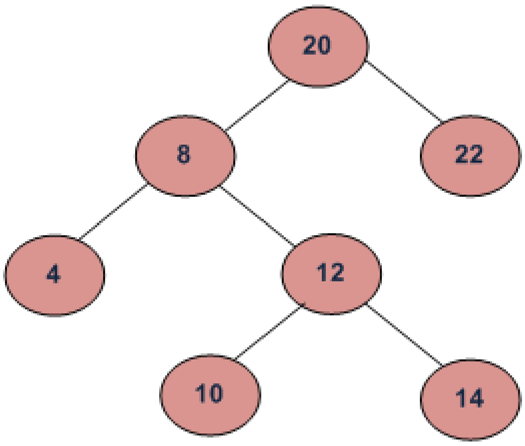
8.Given the head of a singly linked list, return true if it is a palindrome or false otherwise.

Example 1:

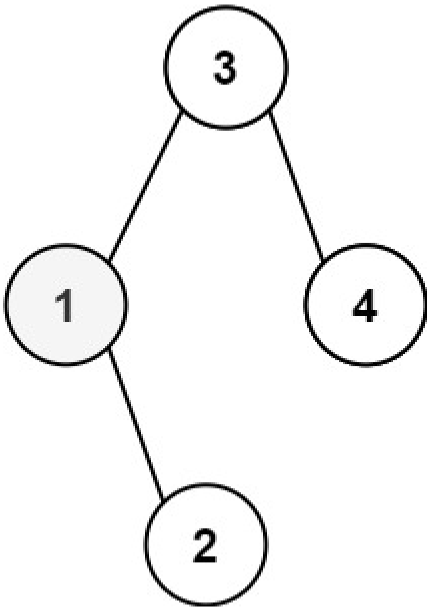
Input: head = [1,2,2,1]

Output: true

9.Given the root of a binary search tree and K as input, find Kth smallest element in BST.
For example, in the following BST,



if k = 3, then the output should be 10, and
if k = 5, then the output should be 14.
Sample:



Input: root = [3,1,4,null,2], k = 1
Output: 1
Input: root = [5,3,6,2,4,null,null,1], k = 3
Output: 3

10. Given a string s, find the frequency of characters
Example 1:
Input: s = "tree"
Output t->1, r->1, e->2

#include <stdio.h>

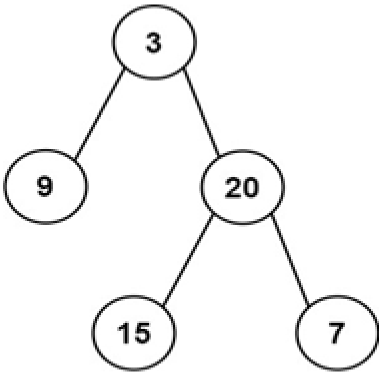
```
#include <stdlib.h>
#include <string.h>
void character_frequency(char* s) {
    int frequency[256] = {0};

    for (int i = 0; i < strlen(s); i++) {
        frequency[(int)s[i]]++;
    }

    for (int i = 0; i < 256; i++) {
        if (frequency[i] > 0) {
            printf("%c -> %d\n", (char)i, frequency[i]);
        }
    }
}

int main() {
    char s[] = "tree";
    character_frequency(s);
    return 0;
}
```

11..Given an unsorted array arr[] with both positive and negative elements, the task is to find the smallest positive number missing from the array.
Input: arr[] = {2, 3, 7, 6, 8, -1, -10, 15}
Output: 1

```
#include<stdio.h>
#include<stdbool.h>
int smallestpositive(int arr[],int n)
{
  bool present[100]={false};
  for (int i=0;i<n;i++)
  {
    if(arr[i]>0&&arr[i]<=100)
    {
      present[arr[i]]=true;
    }
  }
  for(int i=1;i<=100;i++)
  {
    if(!present[i])
    {
      return i;
    }
  }
  return -1;
}
void main()
{
  int arr[]={2,3,7,6,8,-1,-10,15};
  int n=sizeof(arr)/sizeof(arr[0]);
  int result=smallestpositive(arr,n);
  printf("smallest positive number missing is:%d\n",result);
}
```

12.

Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.



Input: preorder = [3,9,20,15,7], inorder = [9,3,15,20,7]
Output: [3,9,20,null,null,15,7]

13.

Write a program to create and display a linked list

Example 1:

Nodes : 6,7,8,9
Output: 6->7->8->9

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
  int data;
  struct Node* next;
} Node;
Node* createNode(int data) {
  Node* newNode = (Node*)malloc(sizeof(Node));
```

```
 newNode->data = data;
 newNode->next = NULL;
 return newNode;
}
void printList(Node* head) {
 while (head != NULL) {
  printf("%d->", head->data);
  head = head->next;
 }
 printf("NULL\n");
}

int main() {
 Node* head = NULL;
 head = createNode(6);
 head->next = createNode(7);
 head->next->next = createNode(8);
 head->next->next->next = createNode(9);

 printf("Linked List: ");
 printList(head);

 return 0;
}
```

Questions

14.

Write a program to sort the below numbers in descending order using bubble sort

Input 4,7,9,1,2

Output:9,7,4,2,1

```
#include<stdio.h>
#include<stdlib.h>
void sort(int a[],int n)
{
        int i,j,temp;
        for(i=0;i<n-1;i++)
        {
                int swap=0;
                for(j=0;j<n-i-1;j++)
                {
                        if(a[j]<a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                                swap=1;
```

```
            }
        }
            if(swap==0)
            break;
    }
    printf("\nsorted array is ");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
}
int main()
{
    int a[10]={4,7,9,1,2},n=6;
    sort(a,n);
 return 0;
}
```

15.

Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.

Input:
N = 5
A[] = {1,2,3,5}

```
#include <stdio.h>
int findMissingElement(int A[], int N) {
  int sumOfN = (N * (N + 1)) / 2;
  int sumOfArray = 0;
  for (int i = 0; i < N - 1; i++) {
    sumOfArray += A[i];
  }
  return sumOfN - sumOfArray;
}
int main() {
  int N = 5;
  int A[] = {1, 2, 3, 5};
  printf("The missing element is: %d\n", findMissingElement(A, N));
  return 0;
}
```

16.

Write a program to find odd number present in the data part of a node

Example Linked List 1->2->3->7
Output: 1,3,7

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void findOddNumbers(struct Node* head) {
    struct Node* current = head;

    printf("Odd numbers in the linked list: ");

    while (current != NULL) {
        if (current->data % 2 != 0) {
            printf("%d ", current->data);
        }
        current = current->next;
    }
    printf("\n");
}
int main() {
    struct Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
    head->next->next->next = createNode(7);
    findOddNumbers(head);
return 0;
}
```

17.Write a program to perform insert and delete operations in a queue
Example : 12,34,56,78
After insertion of 60 content of the queue is 12,34,56,78,60
After deletion of 12 , the contents of the queue : 34,56,78,60

18.Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.
An input string is valid if:
1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

```
Input: s = "()"
Output: true
Input: s = "()[]{}"
Output: true
Input: s = "(]"
Output: false
Input: s = "([)]"
Output: false
Input: s = "{[]}"
Output: true
```

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
#define MAX_SIZE 100
 bool isvalid (char*s)
 {
   char stack[MAX_SIZE];
   int top=-1;
   for (int i=0;i<strlen(s);i++)
   {
     char c=s[i];
     if(c=='('||c=='{'||c=='[')
     {
       stack[++top]=c;
     }
     else
     {
       if(top==-1)return false;
       char openingBracket=stack[top--];
       if((c==')'&&openingBracket!='(')||
         (c=='}'&&openingBracket!='{')||
         (c==']'&&openingBracket!='['))
       {
         return false;
       }
     }
   }
   return top==-1;
 }
void main()
{
  char*s1="(]";
  printf("%s:%s\n",s1,isvalid(s1)?"true":"false");
}
```

20. Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.
Example 1:
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.

21.

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).
Implement the MyQueue class:
1. void push(int x) Pushes element x to the back of the queue.
2. int pop() Removes the element from the front of the queue and returns it.
3. int peek() Returns the element at the front of the queue.
4. boolean empty() Returns true if the queue is empty, false otherwise.
Input
["MyQueue", "push", "push", "peek", "pop", "empty"]
[[], [1], [2], [], [], []]
Output
[null, null, null, 1, 1, false]

Explanation
MyQueue myQueue = new MyQueue();
myQueue.push(1); // queue is: [1]
myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)
myQueue.peek(); // return 1
myQueue.pop(); // return 1, queue is [2]
myQueue.empty(); // return false

Questions
22.

Given an array arr, sort the elements in descending order using bubblesort.

Arr=[9,10,-9,23,67,-90]
Output:[67,23,10,9,-9,-90]

```
#include<stdio.h>
#include<stdlib.h>
void sort(int a[],int n)
{
    int i,j,temp;
    for(i=0;i<n-1;i++)
    {
        int swap=0;
        for(j=0;j<n-i-1;j++)
        {
            if(a[j]<a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                swap=1;
            }
        }
            if(swap==0)
            break;
    }
    printf("\nsorted array is ");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
}
int main()
{
    int a[10]={9,10,-9,23,67,-90},n=6;
    sort(a,n);
  return 0;
}
```

Questions

23. You have been given a positive integer N. You need to find and print the Factorial of this number without using recursion. The Factorial of a positive integer N refers to the product of all number in the range from 1 to N.

Input: N=2

Output: 2

Input: N=4

Output: 24

```
#include<stdio.h>
int main(){
 int num=4;
  unsigned long long factorial=1;
  for(int i=1;i<=num;i++)
  {
    factorial*=i;
  }
  printf("factorial of %d = %d", num, factorial);
  return 0;
}
```

Questions

24.

Given an array arr, sort the elements in ascending order using Bubble sort.

Arr=[9,10,-9,23,67,-90]

Output:[-90,-9,9,10,23,67]

```
#include<stdio.h>
#include<stdlib.h>
void sort(int a[],int n)
{
    int i,j,temp;
    for(i=0;i<n-1;i++)
    {
        int swap=0;
        for(j=0;j<n-i-1;j++)
        {
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                swap=1;
            }
        }
            if(swap==0)
            break;
    }
    printf("\nsorted array is ");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
}
int main()
{
    int a[10]={9,10,-9,23,67,-90};
 int n=6;
    sort(a,n);
 return 0;
}
```

25.

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

1. MinStack() initializes the stack object.
2. void push(int val) pushes the element val onto the stack.
3. void pop() removes the element on the top of the stack.
4. int top() gets the top element of the stack.
5. int getMin() retrieves the minimum element in the stack.

Input

```
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]
```

Output

```
[null,null,null,null,-3,null,0,-2]
```

Explanation

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

26.Find the factorial of a number using iterative procedure

Input : 3

Output: 6

```
#include <stdio.h>
int factorial(int n) {
   int result = 1;
   for (int i = 1; i <= n; i++) {
      result *= i;
   }
   return result;
}

int main() {
   int num=3;
   scanf("%d", &num);
   printf("Factorial of %d is: %d\n", num, factorial(num));
   return 0;
}
```

27.

Given the head of a linked list, insert the node in nth place and return its head.

Input: head = [1,3,2,3,4,5], p=3 n = 2
Output: [1,3,2,3,4,5]
Input: head = [1], p = 0, n = 1
Output: [0,1]
Input: head = [1,2], p=3, n = 3
Output: [1,2,3]

28.

Given the head of a singly linked list and two integers left and right where left <= right, reverse the nodes of the list from position left to position right, and return the reversed list.

Input: head = [1, 2, 3, 4, 5], left = 2, right = 4
Output: [1, 4, 3, 2, 5]

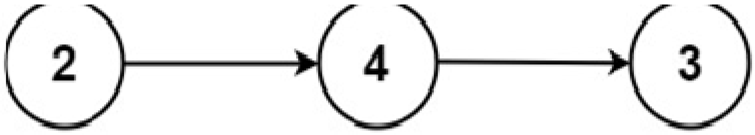
Input: head = [5], left = 1, right = 1
Output: [5]

Input : [10,20,30,40,50,60,70], left = 3, right = 6
Output : [10,20,60,50,40,30,70]

```
#include <stdio.h>
#include <stdlib.h>
struct ListNode {
    int val;
    struct ListNode *next;
};

struct ListNode* reverseBetween(struct ListNode* head, int left, int right) {
    if (!head || left == right) return head;

    struct ListNode dummy;
    dummy.next = head;
    struct ListNode* prev = &dummy;

    for (int i = 1; i < left; i++) {
        prev = prev->next;
    }

    struct ListNode* curr = prev->next;
    struct ListNode* next = NULL;

    for (int i = 0; i < right - left; i++) {
        next = curr->next;
        curr->next = next->next;
        next->next = prev->next;
        prev->next = next;
    }

    return dummy.next;
}
struct ListNode* createNode(int value) {
    struct ListNode* newNode = (struct ListNode*)malloc(sizeof(struct ListNode));
    newNode->val = value;
    newNode->next = NULL;
    return newNode;
}
void printList(struct ListNode* head) {
    while (head) {
        printf("%d -> ", head->val);
        head = head->next;
    }
    printf("NULL\n");
}
int main() {
    struct ListNode* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
    head->next->next->next = createNode(4);
    head->next->next->next->next = createNode(5);

    printf("Original List: ");
    printList(head);

    head = reverseBetween(head, 2, 4);

    printf("Reversed List: ");
    printList(head);

    return 0;
}
```

29.

You are given with the following linked list



The digits are stored in the above order, you are asked to print the list in reverse order.

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

// Define the structure for a linked list node
typedef struct Node {
   int data;
   struct Node* next;
} Node;

// Function to insert a new node at the end of the list
void insert(Node** head, int data) {
   Node* newNode = (Node*) malloc(sizeof(Node));
   newNode->data = data;
   newNode->next = NULL;

   if (*head == NULL) {
      *head = newNode;
   } else {
      Node* temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}

// Function to print the linked list in reverse order
void printReverse(Node* head) {
   if (head == NULL) {
      return;
   }
```

```
    printReverse(head->next);
    printf("%d ", head->data);
}

int main() {
    Node* head = NULL;

    // Insert nodes into the list
    insert(&head, 2);
    insert(&head, 3);
    insert(&head, 6);

    printf("Original list: ");
    Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");

    printf("Reversed list: ");
    printReverse(head);
    printf("\n");

    return 0;
}
```

30.Given two sorted arrays nums1 and nums2 of size m and n respectively, return the sum of these two arrays

Example 1:

Input: nums1 = [1,3], nums2 = [2]

Output: 6

Example 2:

Input: nums1 = [1,2], nums2 = [3,4]

Output: 10

```
#include <stdio.h>
int sumArrays(int nums1[], int m, int nums2[], int n) {
    int sum = 0;
    int i, j;
    for (i = 0; i < m; i++) {
        sum += nums1[i];
    }
    for (j = 0; j < n; j++) {
        sum += nums2[j];
    }
    return sum;
}
```

```
int main() {
   int nums1[] = {1, 3};
   int m = sizeof(nums1) / sizeof(nums1[0]);
   int nums2[] = {2};
   int n = sizeof(nums2) / sizeof(nums2[0]);
   int result = sumArrays(nums1, m, nums2, n);
   printf("The sum of the two arrays is: %d\n", result);
   return 0;
}
```