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

code:

```
#include<stdio.h>
void insertion_sort(int a[],int n)
{
  for(int i=1;i<n-1;i++)
  {
    int j=i-1;
    int temp=a[i];
    while(j>=0 && a[j]>temp)
    {
      a[j+1]=a[j];
      j--;
    }
  a[j+1]=temp;
  }
}
int main()
{
  int a[]={98,23,45,14,6,67,33,42};
  int n=sizeof(a)/sizeof(a[0]);
  insertion_sort(a,n);
  for(int i=0;i<n;i++)
  {
    printf("%d\n",a[i]);
  }
}
```


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

code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node*next;
};
void insert(struct node**head,int data,int *count)
{
  struct node*newnode=(struct node*)malloc(sizeof(struct node));
  newnode->data=data;
  newnode->next=*head;
  *head=newnode;
  (*count)++;
}
void print(struct node*head)
{
struct node *temp=head;
  while(temp->next!=NULL)
  {
    printf("%d-->",temp->data);
    temp=temp->next;
  }
}
int main()
{
  int count=0;
  struct node*head=NULL;
  insert(&head,100,&count);
  insert(&head,60,&count);
  print(head);
  printf(" the count is %d",count);
}
```


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

code:

```
#include<stdio.h>
int fibonacci(int n)
{
  if(n==0)
  {
    return 0;
  }
  else if(n==1)
  {
    return 1;
  }
  return fibonacci(n-1)+fibonacci(n-2);
}
int main()
{
  int n1,p,sum=0;
  printf("enter the size of the array\n");
  scanf("%d",&n1);
  for(int i=0;i<n1;i++)
  {
  p=fibonacci(i);
  printf("%d",p);
  printf("\t");
  sum=sum+fibonacci(i);
  printf("\n");
  }
  printf("the sum is \n");
  printf("%d",sum);
}
```


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

code:

```
#include<stdio.h>
int binary_s(int a[],int n,int search)
{
  int l=0,r=n-1,m;
  while(l<=r)
  {
    m=(l+r)/2;
    if(search==a[m])
    {
      a[m]=search;
      return m;
    }
    else if(search<a[m])
    {
      r=m-1;
    }
    else
    {
      l=m+1;
    }
  }
  return 0;
}
int main()
{
  int a[]={1,5,6,7,9,10};
  int n=sizeof(a)/sizeof(a[0]);
  int s=6;
  int p=binary_s(a,n,s);
  printf("%d",p);
}
```


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

code:

```
#include <stdio.h>
#include <string.h>
void sortString(char* s)
{
    int len = strlen(s);
    for (int i = 0; i < len - 1; i++)
    {
        for (int j = i + 1; j < len; j++)
        {
            if (s[i] > s[j])
            {
                char temp = s[i];
                s[i] = s[j];
                s[j] = temp;
            }
        }
    }
}
int findStartingindex(char* s, char ch)
{
    int index = 0;
    while (s[index] != ch)
    {
        index++;
    }
    return index;
}
int main()
```

```
{
    char s[] = "aaaccc";
    sortString(s);
    printf("Sorted String: %s\n", s);
    for (int i = 0; i < strlen(s); i++)
    {
        if (s[i] == s[i + 1])
        {
            printf("Starting index of repeated character %c: %d\n", s[i], findStartingindex(s, s[i]));
        }
    }
    return 0;
}
```


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

code:

```
#include <stdio.h>
#include <string.h>
void character_frequency(const char *s)
{
    int frequency[256] = {0};
    for (int i = 0; s[i] != '\0'; i++)
    {
        frequency[(unsigned char)s[i]]++;
    }
    for (int i = 0; i < 256; i++)
    {
        if (frequency[i] > 0)
        {
            printf("%c -> %d\n", i, frequency[i]);
        }
    }
}
int main() {
    const 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
Input: arr[] = { 2, 3, -7, 6, 8, 1, -10, 15 }
Output: 4
Input: arr[] = {1, 1, 0, -1, -2}
Output: 2

code:

```
#include<stdio.h>
int find_minimum(int a[],int n)
{
  for(int i=0;i<n;i++)
  {
    while(a[i]>0 && a[i]<=n && a[a[i]-1]!=a[i])
    {
      int temp=a[a[i]-1];
      a[a[i]-1]=a[i];
      a[i]=temp;
    }
  }
for(int i =0;i<n;i++)
{
  if(a[i]!=i+1)
  {
    return i+1;
  }
}
  return n+1;
}
int main()
{
  int a[]={2,3,-7,6,8,1,-10,15};
  int p=sizeof(a)/sizeof(a[0]);
  printf("%d",find_minimum(a,p));
}
```


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

code:

```
#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 printOddNumbers(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);
   printOddNumbers(head);
   return 0;
}
```


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]

code:

```
#include<stdio.h>
void bubble_sort(int a[],int n)
{
  for(int i=0;i<n-1;i++)
  {
    for(int j=0;j<n-1;j++)
    {
      if(a[j+1]<a[j])
      {
        int temp=a[j+1];
        a[j+1]=a[j];
        a[j]=temp;
      }
    }
  }
}
int main()
{
  int a[]={9,10,-9,23,67,-90};
  int n=sizeof(a)/sizeof(a[0]);
  bubble_sort(a,n);
  for(int i=0;i<n;i++)
  {
    printf("%d",a[i]);
    printf("\n");
  }
}
```


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

code:

```
#include<stdio.h>
int main()
{
  int i,n,p=1;
  printf("enter the last number up to factorial will be found ");
  scanf("%d",&n);
 if(n==0)
 {
   return 1;
 }
 for(int i=1;i<=n;i++)
 {
  p=p*i;
 }
  printf("the factorial is\n");
  printf("%d",p);
}
```


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}
Output: 4
Input:
N = 10
A[] = {6,1,2,8,3,4,7,10,5}
Output: 9

code:

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

```
int findMissing(int arr[], int n)
{
    int total = n * (n + 1) / 2;
    int sum = 0;
    for (int i = 0; i < n - 1; i++)
    {
        sum += arr[i];
    }
    return total - sum;
}
int main()
{
    int arr1[] = {1, 2, 3, 5};
    int n1 = 5;
    printf("Missing number: %d\n", findMissing(arr1, n1));
    int arr2[] = {6, 1, 2, 8, 3, 4, 7, 10, 5};
    int n2 = 10;
    printf("Missing number: %d\n", findMissing(arr2, n2));
    return 0;
}
```


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

code:

```
#include <stdio.h>

int sumOfArrays(int nums1[], int m, int nums2[], int n) {
    int sum = 0;
    int merged[m + n];
    int i = 0, j = 0, k = 0;
    while (i < m && j < n) {
        if (nums1[i] < nums2[j]) {
            merged[k++] = nums1[i++];
        } else {
            merged[k++] = nums2[j++];
        }
    }

    while (i < m) {
        merged[k++] = nums1[i++];
    }

    while (j < n) {
        merged[k++] = nums2[j++];
    }

    for (int idx = 0; idx < m + n; idx++) {
        sum += merged[idx];
    }

    return sum;
}
int main() {
    int nums1[] = {1, 2};
    int nums2[] = {3, 4};
    int m = sizeof(nums1) / sizeof(nums1[0]);
    int n = sizeof(nums2) / sizeof(nums2[0]);

    int result = sumOfArrays(nums1, m, nums2, n);
    printf("Sum of the two arrays: %d\n", result);

    return 0;
}
```


--You are given with the following linked list
123

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

code:

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

```
#include <stdlib.h>

struct node {
   int data;
   struct node* next;
};
void insert(struct node** head, int data) {
   struct node* newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = data;
   newnode->next = *head;
   *head = newnode;
}
void printReverse(struct node* head)
{
   if (head == NULL)
   {
      return;
   }
   printReverse(head->next);
   printf("%d-->", head->data);
}
int main() {
   struct node* head = NULL;
   insert(&head, 10);
   insert(&head, 9);
   insert(&head, 7);
   printReverse(head);
   }
```


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]

code:

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

struct node {
    int data;
    struct node* next;
};
void insert(struct node** head, int data) {
    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = data;
    newnode->next = *head;
    *head = newnode;
}
struct node* reverseBetween(struct node* head, int left, int right) {
    if (!head || left == right) return head;

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

    for (int i = 1; i < left; i++) {
        prev = prev->next;
    }
    struct node* current = prev->next;
    struct node* next = NULL;

    for (int i = 0; i < right - left; i++) {
        next = current->next;
        current->next = next->next;
        next->next = prev->next;
        prev->next = next;
    }
    return dummy.next;
}

void printList(struct node* head) {
    while (head) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}
int main() {
    struct node* head = NULL;
    insert(&head, 5);
    insert(&head, 4);
    insert(&head, 3);
    insert(&head, 2);
    insert(&head, 1);
    printf("Original List: ");
    printList(head);
    head = reverseBetween(head, 2, 4);
    printf("Reversed List: ");
    printList(head);

    return 0;
}
```


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]

code:

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

struct Node {
    int data;
    struct Node* next;
};

struct Node* insertNode(struct Node* head, int p, int n) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = p;
    newNode->next = NULL;

    if (n == 0) {
        newNode->next = head;
        return newNode; // New head
    }

    struct Node* current = head;
    for (int i = 0; i < n - 1 && current != NULL; i++) {
        current = current->next;
    }

    if (current == NULL) {
        return head; // Position is out of bounds
    }

    newNode->next = current->next;
    current->next = newNode;

    return head; // Return the original head
}

// Helper function to print the list
void printList(struct Node* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

int main() {
    struct Node* head1 = (struct Node*)malloc(sizeof(struct Node));
    head1->data = 1;
    head1->next = (struct Node*)malloc(sizeof(struct Node));
    head1->next->data = 3;
    head1->next->next = (struct Node*)malloc(sizeof(struct Node));
    head1->next->next->data = 2;
    head1->next->next->next = (struct Node*)malloc(sizeof(struct Node));
    head1->next->next->next->data = 3;
    head1->next->next->next->next = (struct Node*)malloc(sizeof(struct Node));
    head1->next->next->next->next->data = 4;
    head1->next->next->next->next->next = (struct Node*)malloc(sizeof(struct Node));
    head1->next->next->next->next->next->data = 5;
    head1->next->next->next->next->next->next = NULL;

    head1 = insertNode(head1, 3, 2);
    printList(head1); // Output: [1, 3, 2, 3, 4, 5]
}
```


Find the factorial of a number using iterative procedure
Input : 3
Output: 6

code:

```
#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;

    printf("Enter a number: ");
    scanf("%d", &num);

    int fact = factorial(num);

    printf("The factorial of %d is %d\n", num, fact);

    return 0;
}
```


De