**Exercise 1**

**1.** **#include <bits/stdc++.h>**

**using namespace std;**

**struct Node {**

**int key;**

**struct Node\* next;**

**};**

**Node\* newNode(int key)**

**{**

**Node\* temp = new Node;**

**temp->key = key;**

**temp->next = NULL;**

**return temp; }**

**// A utility function to print a linked list**

**void printList(Node\* head)**

**{**

**while (head != NULL) {**

**cout << head->key << " ";**

**head = head->next;**

**}**

**cout << endl;**

**}**

**// Function to detect first node of loop**

**// in a linked list that may contain loop**

**bool detectLoop(Node\* head)**

**{ // Create a temporary node**

**Node\* temp = new Node;**

**while (head != NULL) {**

**// This condition is for the case**

**// when there is no loop**

**if (head->next == NULL) {**

**return false;**

**}**

**// Check if next is already**

**// pointing to temp**

**if (head->next == temp) {**

**return true;**

**}**

**// Store the pointer to the next node in order to get to it in the next step**

**Node\* nex = head->next;**

**// Make next point to temp**

**head->next = temp;**

**// Get to the next node in the list**

**head = nex; }**

**return false;**

**}**

**/\* Driver program to test above function\*/**

**int main()**

**{**

**Node\* head = newNode(3);**

**head->next = newNode(2);**

**head->next->next = newNode(0);**

**head->next->next->next = newNode(-4);**

**// head->next->next->next->next = newNode(5);**

**/\* Create a loop for testing(-4 is pointing to 2) \*/**

**head->next->next->next = head->next;**

**bool found = detectLoop(head);**

**if (found)**

**cout << "true";**

**else**

**cout << "false";**

**return 0;**

**}**

**2.**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node {**

**int key;**

**struct Node\* next;**

**};**

**Node\* newNode(int key)**

**{**

**Node\* temp = new Node;**

**temp->key = key;**

**temp->next = NULL;**

**return temp;**

**}**

**// A utility function to print a linked list**

**void printList(Node\* head)**

**{**

**while (head != NULL) {**

**cout << head->key << " ";**

**head = head->next; }**

**cout << endl; }**

**// Function to detect first node of loop in a linked list that may contain loop**

**bool detectLoop(Node\* head)**

**{**

**// Create a temporary node**

**Node\* temp = new Node;**

**while (head != NULL) {**

**// This condition is for the case**

**// when there is no loop**

**if (head->next == NULL) {**

**return false; }**

**// Check if next is already pointing to temp**

**if (head->next == temp) {**

**return true; }**

**// Store the pointer to the next node in order to get to it in the next step**

**Node\* nex = head->next;**

**// Make next point to temp**

**head->next = temp;**

**// Get to the next node in the list**

**head = nex; }**

**return false;**

**}**

**/\* Driver program to test above function\*/**

**int main()**

**{**

**Node\* head = newNode(1);**

**head->next = newNode(2);**

**/\* Create a loop for testing(2 is pointing to 1) \*/**

**head->next = head;**

**bool found = detectLoop(head);**

**if (found)**

**cout << "true";**

**else**

**cout << "false";**

**return 0;**

**}**

**3.** **#include <bits/stdc++.h>**

**using namespace std;**

**struct Node {**

**int key;**

**struct Node\* next;**

**};**

**Node\* newNode(int key)**

**{**

**Node\* temp = new Node;**

**temp->key = key;**

**temp->next = NULL;**

**return temp;**

**}**

**void printList(Node\* head)**

**{**

**while (head != NULL) {**

**cout << head->key << " ";**

**head = head->next;**

**}**

**cout << endl;**

**}**

**bool detectLoop(Node\* head)**

**{**

**// Create a temporary node**

**Node\* temp = new Node;**

**while (head != NULL) {**

**if (head->next == NULL) {**

**return false;**

**}**

**if (head->next == temp) {**

**return true;**

**}**

**Node\* nex = head->next;**

**head->next = temp;**

**head = nex; }**

**return false;**

**}**

**int main()**

**{**

**Node\* head = newNode(1);**

**bool found = detectLoop(head);**

**if (found)**

**cout << "true";**

**else**

**cout << "false";**

**return 0;**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1.WAP to Check Prime Number using loop?**

**#include<iostream>**

**using namespace std;**

**int main(){**

**int n,i=2;**

**cin>>n;**

**while(i<=n/2){**

**if(n%i==0)**

**break;**

**i++;**

**}**

**if(i<=n/2)**

**cout<<"Not Prime"<<endl;**

**else**

**cout<<"Prime"<<endl;**

**return 0;**

**}**

**----------------------------------------------------------------------------------------------------------------**

**2.WAP to Check if a Number is Positive or Negative**

**#include<iostream>**

**using namespace std;**

**int main ()**

**{**

**int num;**

**cout << "Enter the number to be checked : ";**

**cin >> num;**

**if (num >= 0)**

**cout << num << " is a positive number.";**

**else**

**cout << num << " is a negative number.";**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**3.WAP to Find Factorial of a number**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int i,fact=1,number;**

**cout<<"Enter any Number: ";**

**cin>>number;**

**for(i=1;i<=number;i++){**

**fact=fact\*i;**

**}**

**cout<<"Factorial of " <<number<<" is: "<<fact<<endl;**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**4. WAP to Display Fibonacci series**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n1=0,n2=1,n3,i,number;**

**cout<<"Enter the number of elements: ";**

**cin>>number;**

**cout<<n1<<" "<<n2<<" ";**

**for(i=2;i<number;++i)**

**{**

**n3=n1+n2;**

**cout<<n3<<" ";**

**n1=n2;**

**n2=n3;**

**}**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**5. WAP to Display Fibonacci series up to a given**

**number (instead of terms)**

**Input: 20**

**Output:0 + 1 + 1 + 2 + 3 + 5 + 8 + 13**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int limit, first=0, second=1, next, num;**

**cout <<"Enter the limit of Fibonacci series"<<endl;**

**cin >> num;**

**for(int p=0;p<num;p++)**

**{**

**if (p <= 1)**

**next = p;**

**else**

**{**

**next = first + second;**

**first = second;**

**second = next;**

**}**

**cout<<next<<" ";**

**}**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

1. **WAP to Display Uppercased A to Z**

**void uppercaseAlphabets()**

**{**

**for (char c = 'A'; c <= 'Z'; ++c)**

**cout << c << " ";**

**cout << endl;**

**}**

**int main()**

**{**

**cout << "Uppercase Alphabets" << endl;**

**uppercaseAlphabets(ch);**

**cout << "Lowercase Alphabets " << endl;**

**lowercaseAlphabets(ch);**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

1. **WAP to Swap two numbers using temporary variable**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int a = 5, b = 10, temp;**

**cout <<"Before swapping."<<endl;**

**cout <<"a ="<<a<<",b ="<<b<<endl;**

**temp = a;**

**a = b;**

**b = temp;**

**cout << "\nAfter swapping." << endl;**

**cout << "a = "<< a << ", b = "<< b << endl;**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**8. WAP to Swap two numbers without using temporary**

**Variable**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int a=5, b=10;**

**cout<<"Before swap a= "<<a<<" b= "<<b<<endl;**

**a=a\*b;**

**b=a/b;**

**a=a/b;**

**cout<<"After swap a= "<<a<<" b= "<<b<<endl;**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**9. WAP to Check whether an alphabet is vowel or**

**Consonant**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**char c;**

**int isLowercaseVowel, isUppercaseVowel;**

**cout << "Enter an alphabet: ";**

**cin >> c;**

**isLowercaseVowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');**

**isUppercaseVowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');**

**if (isLowercaseVowel || isUppercaseVowel)**

**cout << c << " is a vowel.";**

**else**

**cout << c << " is a consonant.";**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**10. WAP to Find the largest number among the given**

**Numbers**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int a=3,b=4,c=10;**

**if(a>b && a>c){**

**cout<<"is greatest among three numbers"<<a;**

**}**

**else if(b>a && b>c){**

**cout<<"is greatest among three numbers"<<b;**

**}**

**else if(c>a && c>b){**

**cout<< "is greatest among three numbers"<<c;**

**}**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**11. WAP to Reverse a Number**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n, reversedNumber = 0, remainder;**

**cout << "Enter an integer:";**

**cin >> n;**

**while(n != 0) {**

**remainder = n%10;**

**reversedNumber = reversedNumber\*10 + remainder;**

**n /= 10;**

**}**

**cout << "Reversed Number ="<< reversedNumber;**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**12. WAP to Program to Check Palindrome**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int n,r,sum=0,temp;**

**cout<<"Enter the Number=";**

**cin>>n;**

**temp=n;**

**while(n>0)**

**{**

**r=n%10;**

**sum=(sum\*10)+r;**

**n=n/10;**

**}**

**if(temp==sum)**

**cout<<"Number is Palindrome.";**

**else**

**cout<<"Number is not Palindrome.";**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**13. WAP to Program to Check Prime Number**

**---------------------------------------------------------------------------------------------------------------**

**14. WAP to Display Prime Numbers Between two**

**Intervals**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int low, high, i;**

**bool isPrime = true;**

**cout << "Enter two numbers (intervals): ";**

**cin >> low >> high;**

**cout << "\nPrime numbers between " << low << " and " << high << " are: " << endl;**

**while (low < high) {**

**isPrime = true;**

**if (low == 0 || low == 1) {**

**isPrime = false;**

**}**

**else {**

**for (i = 2; i <= low / 2; ++i) {**

**if (low % i == 0) {**

**isPrime = false;**

**break;**

**}**

**}**

**}**

**if (isPrime)**

**cout << low << " ";**

**++low;**

**}**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**15. WAP to Check Armstrong Number for 3 digit**

**Number**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int num, originalNum, remainder, result = 0;**

**cout << "Enter a three-digit integer: ";**

**cin >> num;**

**originalNum = num;**

**while (originalNum != 0) {**

**remainder = originalNum % 10;**

**result += remainder \* remainder \* remainder;**

**originalNum /= 10;**

**}**

**if (result == num)**

**cout << num << " is an Armstrong number.";**

**else**

**cout << num << " is not an Armstrong number.";**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**16. WAP to Check Armstrong Number for n digit**

**Number**

**#include <cmath>**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int num, originalNum, remainder, n = 0, result = 0, power;**

**cout << "Enter an integer: ";**

**cin >> num;**

**originalNum = num;**

**while (originalNum != 0) {**

**originalNum /= 10;**

**++n;**

**}**

**originalNum = num;**

**while (originalNum != 0) {**

**remainder = originalNum % 10;**

**power = round(pow(remainder, n));**

**result += power;**

**originalNum /= 10;**

**}**

**if (result == num)**

**cout << num << " is an Armstrong number.";**

**else**

**cout << num << " is not an Armstrong number.";**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**17. WAP to Armstrong Numbers Between Two**

**Integers**

**#include <stdio.h>**

**#include <math.h>**

**int main() {**

**int low = 100;**

**int high = 400;**

**printf("The amstrong numbers between %d and %d is \n",low,high);**

**for (int i = low+1; i < high; ++i) {**

**int x = i;**

**int n = 0;**

**while (x != 0) {**

**x /= 10;**

**++n;**

**}**

**int pow\_sum = 0;**

**x = i;**

**while (x != 0) {**

**int digit = x % 10;**

**pow\_sum += pow(digit, n);**

**x /= 10;**

**}**

**if (pow\_sum == i)**

**printf("%d ", i);**

**}**

**printf("\n");**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**18. WAP to Prime Numbers Between Two Integers**

**#include <iostream>**

**using namespace std;**

**int checkPrimeNumber(int);**

**int main() {**

**int n1, n2;**

**bool flag;**

**cout << "Enter two positive integers: ";**

**cin >> n1 >> n2;**

**if (n1 > n2) {**

**n2 = n1 + n2;**

**n1 = n2 - n1;**

**n2 = n2 - n1;**

**}**

**cout << "Prime numbers between " << n1 << " and " << n2 << " are: ";**

**for(int i = n1+1; i < n2; ++i) {**

**flag = checkPrimeNumber(i);**

**if(flag)**

**cout << i << " ";**

**}**

**return 0;**

**}**

**int checkPrimeNumber(int n) {**

**bool isPrime = true;**

**if (n == 0 || n == 1) {**

**isPrime = false;**

**}**

**else {**

**for(int j = 2; j <= n/2; ++j) {**

**if (n%j == 0) {**

**isPrime = false;**

**break;**

**}**

**}**

**}**

**return isPrime;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**19. WAP to Represent a number as Sum of Two Prime**

**Numbers**

**Input: 34 Output: 34 = 3 + 31, 34 = 5 + 29, 34 = 11 +**

**23, 34 = 17 + 17**

**#include <iostream>**

**using namespace std;**

**int checkPrime(int n);**

**int main() {**

**int n, i, flag = 0;**

**cout<<"Enter a positive integer";**

**cin>>n;**

**for (i = 2; i <= n / 2; ++i) {**

**if (checkPrime(i) == 1) {**

**if (checkPrime(n - i) == 1) {**

**printf("%d = %d + %d\n", n, i, n - i);**

**flag = 1;**

**}**

**}**

**}**

**if (flag == 0)**

**cout<<" cannot be expressed as the sum of two prime numbers."<< n;**

**return 0;**

**}**

**int checkPrime(int n) {**

**int i, isPrime = 1;**

**for (i = 2; i <= n / 2; ++i) {**

**if (n % i == 0) {**

**isPrime = 0;**

**break;**

**}**

**}**

**return isPrime;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**20. WAP to Convert Decimal to Octal**

**#include <iostream>**

**using namespace std;**

**int decimalToOctal(int decimalnum)**

**{**

**int octalnum = 0, temp = 1;**

**while (decimalnum != 0)**

**{**

**octalnum = octalnum + (decimalnum % 8) \* temp;**

**decimalnum = decimalnum / 8;**

**temp = temp \* 10;**

**}**

**return octalnum;**

**}**

**int main()**

**{**

**int decimalnum;**

**cout<<"Enter a Decimal Number";**

**cin>>decimalnum;**

**cout<<"Equivalent Octal Number"<<decimalToOctal(decimalnum);**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**21. WAP to Convert Octal to Decimal**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**long int octal, decimal = 0;**

**int i = 0;**

**cout<<"Enter any octal number";**

**cin>>octal;**

**while (octal != 0)**

**{**

**decimal = decimal +(octal % 10)\* pow(8, i++);**

**octal = octal / 10;**

**}**

**cout<<"Equivalent decimal value:"<<decimal;**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**22. WAP to convert binary number to decimal**

**#include <iostream>**

**using namespace std;**

**int convert(long long n);**

**int main() {**

**long long n;**

**cout<<"Enter a binary number:";**

**cin>>n;**

**printf("%lld in binary = %d in decimal", n, convert(n));**

**return 0;**

**}**

**int convert(long long n) {**

**int dec = 0, i = 0, rem;**

**while (n != 0) {**

**rem = n % 10;**

**n /= 10;**

**dec += rem \* pow(2, i);**

**++i;**

**}**

**return dec;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**23. WAP to convert decimal number to binary**

**#include <iostream>**

**using namespace std;**

**void decToBinary(int n)**

**{**

**int binaryNum[32];**

**int i = 0;**

**while (n > 0) {**

**binaryNum[i] = n % 2;**

**n = n / 2;**

**i++;**

**}**

**for (int j = i - 1; j >= 0; j--)**

**cout << binaryNum[j];**

**}**

**int main()**

**{**

**int n = 17;**

**decToBinary(n);**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**24. WAP to Factorial of a Number Using Recursion**

**#include<stdio.h>**

**using namespace std;**

**long int multiplyNumbers(int n);**

**int main() {**

**int n;**

**printf("Enter a positive integer: ");**

**scanf("%d",&n);**

**printf("Factorial of %d = %ld", n, multiplyNumbers(n));**

**return 0;**

**}**

**long int multiplyNumbers(int n) {**

**if (n>=1)**

**return n\*multiplyNumbers(n-1);**

**else**

**return 1;**

**}**

**25. WAP to Convert Decimal to Octal**

**#include <iostream>**

**using namespace std;**

**void decToOctal(int n)**

**{**

**int octalNum[100];**

**int i = 0;**

**while (n != 0) {**

**octalNum[i] = n % 8;**

**n = n / 8;**

**i++;**

**}**

**for (int j = i - 1; j >= 0; j--)**

**cout << octalNum[j];**

**}**

**int main()**

**{**

**int n = 33;**

**decToOctal(n);**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**26. WAP to Convert Binary to Octal**

**#include <stdio.h>**

**using namespace std;**

**int main()**

**{**

**long int binarynum, octalnum = 0, j = 1, remainder;**

**printf("Enter the value for binary number: ");**

**scanf("%ld", &binarynum);**

**while (binarynum != 0)**

**{**

**remainder = binarynum % 10;**

**octalnum = octalnum + remainder \* j;**

**j = j \* 2;**

**binarynum = binarynum / 10;**

**}**

**printf("Equivalent octal value: %lo", octalnum);**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**27. WAP to Convert Octal to Binary**

**#include <stdio.h>**

**#include <math.h>**

**long octalToBinary(int octalnum)**

**{**

**int decimalnum = 0, i = 0;**

**long binarynum = 0;**

**while(octalnum != 0)**

**{**

**decimalnum = decimalnum + (octalnum%10) \* pow(8,i);**

**i++;**

**octalnum = octalnum / 10;**

**}**

**i = 1;**

**while (decimalnum != 0)**

**{**

**binarynum = binarynum + (decimalnum % 2) \* i;**

**decimalnum = decimalnum / 2;**

**i = i \* 10;**

**}**

**return binarynum;**

**}**

**int main()**

**{**

**int octalnum;**

**printf("Enter an octal number: ");**

**scanf("%d", &octalnum);**

**printf("Equivalent binary number is: %ld", octalToBinary(octalnum));**

**return 0;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**28. WAP to Reverse a Sentence Using Recursion**

**Input : CDAC Mumbai**

**Output: iabmum CADC**

**#include <stdio.h>**

**using namespace std;**

**void reverseSentence();**

**int main() {**

**printf("Enter a sentence: ");**

**reverseSentence();**

**return 0;**

**}**

**void reverseSentence() {**

**char c;**

**scanf("%c", &c);**

**if (c != '\n') {**

**reverseSentence();**

**printf("%c", c);**

**}**

**}**

**---------------------------------------------------------------------------------------------------------------**

**29. WAP to calculate power using recursion**

**#include <stdio.h>**

**using namespace std;**

**int power(int n1, int n2);**

**int main() {**

**int base, a, result;**

**printf("Enter base number: ");**

**scanf("%d", &base);**

**printf("Enter power number(positive integer): ");**

**scanf("%d", &a);**

**result = power(base, a);**

**printf("%d^%d = %d", base, a, result);**

**return 0;**

**}**

**int power(int base, int a) {**

**if (a != 0)**

**return (base \* power(base, a - 1));**

**else**

**return 1;**

**}**

**---------------------------------------------------------------------------------------------------------------**

**30. WAP to Find the largest element in an array**

**Input: 89, 34 50, 23, 100, 39,455**

**Output: 455**

**#include <stdio.h>**

**using namespace std;**

**int main()**

**{**

**int size, i, largest;**

**printf("\n Enter the size of the array: ");**

**scanf("%d", &size);**

**int array[size];**

**printf("\n Enter %d elements of the array: \n", size);**

**for (i = 0; i < size; i++)**

**{**

**scanf("%d", &array[i]);**

**}**

**largest = array[0];**

**for (i = 1; i < size; i++)**

**{**

**if (largest < array[i])**

**largest = array[i];**

**}**

**printf("\n largest element present in the given array is : %d", largest);**

**return 0;**

**}**