**Lab Assignment – 01**

**ID: C221060**

**(01)**

**/// Write a program to count number of significant digits in a given number.**

**/// C221060**

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

**using namespace std;**

**void solve()**

**{**

**string number;**

**cin >> number;**

**deque<char>integer,decimal;**

**bool point = false;**

**for(auto a : number){**

**if(a == '.'){**

**point = true;**

**break;**

**}**

**else{**

**integer.push\_back(a);**

**}**

**}**

**if(point){**

**string t = number;**

**reverse(t.begin(),t.end());**

**for(auto a : t){**

**if(a == '.') break;**

**decimal.push\_front(a);**

**}**

**while(integer.size() != 0 and integer.front() == '0'){**

**integer.pop\_front();**

**}**

**if(integer.size() == 0){**

**while(decimal.size() != 0 and decimal.front() == '0'){**

**decimal.pop\_front();**

**}**

**cout << "Total number of Significant digits: " << ( decimal.size() ) << endl;**

**}**

**else{**

**cout << "Total number of Significant digits: " << ( integer.size() + decimal.size() ) << endl;**

**}**

**}**

**else{**

**while(integer.size() != 0 and integer.front() == '0'){**

**integer.pop\_front();**

**}**

**while(integer.size() != 0 and integer.back() == '0'){**

**integer.pop\_back();**

**}**

**cout << "Total number of Significant digits: " << integer.size() << endl;**

**}**

**}**

**int main()**

**{**

**int t;**

**cin >> t;**

**while(t--)**

**{**

**solve();**

**}**

**return 0;**

**}**

**/\***

**5**

**00200**

**200**

**002.00200**

**0.00**

**0.00300**

**\*/**

**(02)**

**/// Write a program to round off a number with n significant figures using banker’s rule.**

**/// C221060**

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

**using namespace std;**

**void solve()**

**{**

**string number;**

**cin >> number;**

**int digit;**

**cin >> digit;**

**int point;**

**for(int i = 0; i < number.size(); i++){**

**if(number[i] == '.'){**

**point = i;**

**break;**

**}**

**}**

**if( (number[point + digit + 1] < '5') || (number[point + digit + 1] == '5' and (number[point + digit] - '0') % 2 == 0 )){**

**cout << "The answer is: ";**

**for(int i = 0; i <= point + digit; i++){**

**cout << number[i];**

**}**

**cout << endl;**

**}**

**else{**

**bool f = false,inc = false;**

**if(number[point + digit] != '9') f = true;**

**string ans = "";**

**if(f){**

**number[point + digit] = char(number[point + digit] + 1);**

**cout << "The answer is: ";**

**for(int i = 0; i <= point + digit; i++){**

**cout << number[i];**

**}**

**cout << endl;**

**}**

**else{**

**for(int i = point + digit; i >= 0; i--){**

**if(number[i] != '9' and number[i] != '.') f = true;**

**if(number[i] == '.'){**

**ans += number[i];**

**continue;**

**}**

**if(f == true and inc == true){**

**ans += number[i];**

**}**

**else if(f == false and inc == false){**

**ans += '0';**

**}**

**else if(f == true and inc == false){**

**ans += char(number[i] + 1);**

**inc = true;**

**}**

**}**

**if(f == false){**

**ans += '1';**

**}**

**reverse(ans.begin(),ans.end());**

**cout << "The answer is: ";**

**cout << ans << endl;**

**}**

**}**

**}**

**int main()**

**{**

**int t;**

**cin >> t;**

**while(t--)**

**{**

**solve();**

**}**

**return 0;**

**}**

**/\***

**5**

**2.9956**

**3**

**2.9966**

**3**

**99.99999**

**3**

**98.99999**

**3**

**5.9996**

**3**

**\*/**

**(03)**

**/// Write a program to evaluate a polynomial f(x) = x3 - 2x2 + 5x + 10 by using Horner's rule x = 5.**

**/// C221060**

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

**using namespace std;**

**int main()**

**{**

**cout << "Enter the value of n = ";**

**int n,x;**

**cin >> n;**

**vector<int> a(n+5),p(n+5);**

**for(int i = n; i >= 0; i--){**

**cout << "a[" << i << "] = ";**

**cin >> a[i];**

**}**

**cout << "Value of x = ";**

**cin >> x;**

**p[n+1] = 0;**

**for(int i = n; i >= 0; i--){**

**p[i] = (p[i+1] \* x) + a[i];**

**}**

**cout << "The answer is = " << p[0] << endl;**

**return 0;**

**}**

**(04)**

**/// Write a program to find the root of the equation x3 – 9x + 1 = 0, correct to 3 decimal places, by using the bisection method.**

**/// C221060**

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

**using namespace std;**

**double error = .0001;**

**bool calc(double x)**

**{**

**double xx = ( (x \* x \* x) - (9.0 \* x) + 1.0 );**

**if(xx < 0.0) return true;**

**else return false;**

**}**

**int main()**

**{**

**double boro = 3.0, choto = 1.0;**

**while(fabs(boro - choto) >= error)**

**{**

**double mid = (boro + choto) / 2.0;**

**if(calc(mid)){**

**choto = mid;**

**}**

**else boro = mid;**

**}**

**cout << fixed << setprecision(3) << choto << endl;**

**return 0;**

**}**

**(05)**

**/// Write a program to find all the roots of the equation x3 - 6x + 4 = 0, correct to 3 decimal places. [Use bisection method].**

**/// C221060**

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

**using namespace std;**

**double error = 0.0001;**

**double fun(double x)**

**{**

**return 1.00 \* ( (x \* x \*x) - (6 \* x) + 4);**

**}**

**double bisection\_root(double x1, double x2)**

**{**

**while(fabs(x1 - x2) > error)**

**{**

**double mid = ( x1 + x2) / 2.00;**

**if(fun(mid) \* fun(x1) < 0.0)**

**x2 = mid;**

**else**

**x1 = mid;**

**}**

**return x2;**

**}**

**int main()**

**{**

**double lower = -100, upper = 100, x = 1.0; ///boundary and increment**

**double x2 = lower, x1 = lower;**

**while(x2 < upper)**

**{**

**x1 = lower, x2 = lower + x;**

**double f1 = fun(x1), f2 = fun(x2);**

**lower = x2 + 0.1;**

**if((f1 \* f2) > 0)**

**{**

**continue;**

**}**

**cout << bisection\_root(x1, x2) << endl;**

**}**

**}**

**(06)**

**/// Write a program to find the root of the equation x3 - 6x + 4 = 0, correct to 3 decimal places, by using Newton-Raphson method.**

**/// C221060**

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

**using namespace std;**

**/\***

**Newton-Raphson Method**

**Example: Given, x^3 - 6x + 4 = 0**

**Let, f(x) = x^3 - 6x + 4**

**Thus, f'(x) = 3x^2 - 6 (first derivative)**

**Fix two point, x1, and x2**

**Assume x1 initially and find x2 by the formula,**

**x2 = x1 - f(x1) / f'(x1)**

**Replace x1 by x2 and find x2 again,**

**Repeat this process until fabs(x2 -x1 ) < E**

**\*/**

**double error = .005;**

**/// Finding f(x1)**

**double f\_x1(double x1)**

**{**

**return (x1 \* x1 \* x1) - (6 \* x1) + 4;**

**}**

**/// Finding f'(x1)**

**double f\_prime(double x1)**

**{**

**return (3 \* x1 \* x1) - 6.0;**

**}**

**int main()**

**{**

**double x1 = 0; /// Assumption**

**double x2 = x1 - ( f\_x1(x1) / f\_prime(x1) );**

**while( fabs(x2 - x1) > error )**

**{**

**x1 = x2;**

**x2 = x1 - ( f\_x1(x1) / f\_prime(x1) );**

**}**

**cout << "The result is: " << fixed << setprecision(4) << x1 << endl;**

**return 0;**

**}**

**(07)**

**/// Write a program to find the root of the equation x3 - x + 2 = 0, correct to 3 decimal places, by using false position method.**

**/// C221060**

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

**using namespace std;**

**/\***

**False Position method**

**For root point(x,y), x = x0 and y = 0;**

**At first, we should decide initial value of x1 and x2**

**By placing root point to line equation ((f (x2) – f (x1)) / (x2 - x1) = (y- f (x1)) / (x - x1))**

**we find,**

**x0 = x1- (f(x1) (x2-x1) ) / (f (x2) – f (x1))**

**If f (x0) \* f (x1) < 0 set x2 = x0 otherwise set x1 = x0**

**Repeat until the absolute difference between two successive x0 is less then E**

**\*/**

**double error = .00005;**

**double f\_x(double x)**

**{**

**return ((x \* x \* x) - x + 2) \* 1.00;**

**}**

**int main()**

**{**

**double x1 = -2.0, x2 = 1.0;**

**double x0 = x1 - ( ( f\_x(x1) \* (x2- x1) ) / ( f\_x(x2) - f\_x(x1) ) );**

**if( f\_x(x1) \* f\_x(x0) < 0.0 ) x2 = x0;**

**else x1 = x0;**

**double x0\_prev = x0;**

**x0 = x1 - ( ( f\_x(x1) \* (x2- x1) ) / ( f\_x(x2) - f\_x(x1) ) );**

**while( abs(x0\_prev - x0) > error )**

**{**

**if( f\_x(x1) \* f\_x(x0) < 0.0 ) x2 = x0;**

**else x1 = x0;**

**x0\_prev = x0;**

**x0 = x1 - ( ( f\_x(x1) \* (x2- x1) ) / ( f\_x(x2) - f\_x(x1) ) );**

**}**

**cout << fixed << setprecision(4) << x0 << endl;**

**return 0;**

**}**

**(08)**

**/// Write a program to find the root of the equation x3 – 5x2 –29 = 0, correct to 3 decimal places, by using secant method.**

**/// C221060**

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

**using namespace std;**

**double error = 0.0001;**

**double fun(double x)**

**{**

**return ( (x \* x \* x) - (5 \* x \* x) - 29 );**

**}**

**double calc(double x0, double x1)**

**{**

**return x1 - ( ( (x1 - x0) / (fun(x1) - fun(x0)) ) \* fun(x1) );**

**}**

**int main()**

**{**

**double x0 = 1, x1 = 6;**

**while(fabs(x0 - x1) >= error)**

**{**

**double x2 = calc(x0,x1);**

**x0 = x1;**

**x1 = x2;**

**}**

**cout << fixed << setprecision(3) << x1 << endl;**

**return 0;**

**}**

**(09)**

**/// Write a program to find the quotient polynomial q(x) such that p(x) = (x - 2) q(x) where the polynomial p(x) = x3 - 5x2 + 10x - 8 = 0 has a root at x = 2.**

**/// C221060**

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

**using namespace std;**

**int main()**

**{**

**int n, x;**

**cout << "n = ";**

**cin >> n;**

**cout << "x = ";**

**cin >> x;**

**vector<int>a(n+5),b(n+5);**

**for(int i = n; i>=0; i--)**

**{**

**cout << "a[" << i << "] = ";**

**cin >> a[i];**

**}**

**b[n] = 0;**

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

**b[i] = a[i+1] + (x \* b[i+1]);**

**}**

**cout << "\nThe Quotient Polynomial q(x) is, \n";**

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

**if(b[i] == 0){**

**continue;**

**}**

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

**cout << b[i] << "x^" << i << " ";**

**continue;**

**}**

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

**if(b[i] >= 0) cout << "+ " << b[i] << " ";**

**else cout << "- " << abs(b[i]) << " ";**

**continue;**

**}**

**else{**

**if(b[i] >= 0 ) cout << "+ " << b[i] << "x^" << i << " ";**

**else cout << "- " << abs(b[i]) << "x^" << i << " ";**

**}**

**}**

**cout << "= 0" << endl;**

**return 0;**

**}**

**/\***

**input:**

**3**

**2**

**1**

**-5**

**10**

**8**

**output:**

**The Quotient Polynomial q(x) is,**

**1x^2 - 3x^1 + 4 = 0**

**\*/**

**(10)**

**/\***

**Write a program to find all the roots of the equation x3 - 6x + 4 = 0,**

**correct to 3 decimal places. [Use Newton-Raphson method with deflation].**

**\*/**

**/// C221060**

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

**using namespace std;**

**double error = 0.0001;**

**double fx(double x, vector<double>a)**

**{**

**int n = a.size();**

**double result = 0.0;**

**for(int i = n; i >= 0; i--){**

**result += (1.0 \* a[i] \* pow(x,i));**

**}**

**return result;**

**}**

**double dfx(double x, vector<double>a)**

**{**

**int n = a.size();**

**double result = 0.0;**

**for(int i = n; i >= 1; i--){**

**result += (1.0 \* a[i] \* (pow(x,i-1)));**

**}**

**return result;**

**}**

**int main()**

**{**

**int n;**

**cout << "n = ";**

**cin >> n;**

**vector<double> a(n+1);**

**for(int i = n; i >= 0; i--){**

**cout << "a[" << i << "] = ";**

**cin >> a[i];**

**}**

**vector<double> ans;**

**double x0 = 1.0;**

**while(n > 1){**

**double x1 = x0;**

**double x2 = x1 - ( fx(x1, a) / dfx(x1, a));**

**while(fabs(x2 - x1) > error){**

**x1 = x2;**

**x2 = x1 - ( fx(x1, a) / dfx(x1, a));**

**}**

**ans.push\_back(x2);**

**vector<double>b = a;**

**b[n] = 0;**

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

**b[i] = a[i+1] + (x2 \* b[i+1]);**

**}**

**b.pop\_back();**

**a = b;**

**n--;**

**}**

**ans.push\_back(-a[0]/a[1]);**

**for(int i = 0; i < ans.size(); i++){**

**cout << "Root " << (i+1) << " : " << fixed << setprecision(3) << ans[i] << endl;**

**}**

**return 0;**

**}**

**/\***

**input:**

**3**

**1**

**0**

**-6**

**4**

**output:**

**Root 1 : 0.732**

**Root 2 : 2.000**

**Root 3 : -2.732**

**\*/**