Tutorial & Laboratory

Programming & Data Structure: CS11001/19001

Section - 4/D

DO NOT POWER ON THE MACHINE

Department of Computer Science and Engineering

I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (**06.02.2014**)

Download

Download the file date060214.pdf from Programming & Data Structures ... of

http://cse.iitkgp.ac.in/~goutam

View the file using the command acroread & or xpdf &

Laboratory Test I : 13th February, 2014

- Up to iteration and function (non-recursive).
- Book, note book, and internet resources are not allowed.
- Time: Two hours.
- Marks: 25

The test will start after the tutorial.

Assignment VI

Write a C program that reads a set of n ($\leq 10^5$) positive integers in a 1-D array `a[]` and performs the following four tasks.
[**Marks:** $4 + 6 + 5 + 5$]

Task I

Print n data from `a[]` and also copy the elements of the `a[]` to another 1-D array `b[]` in reverse order i.e.
`b[i]` $\leftarrow$ `a[n-(i+1)]`, $i = 0, 1, \cdots, n-1$. Print the content of `b[]`.
Example:

```
a[]: 23 10 45 7 1
b[]: 1 7 45 10 23
```

Task II

Print only those data from `a[]` that are Fibonacci numbers. Also print the total count of such data.

Input: `5`

Input: `233 87 13 6763 1597`

Output: `233 13 1597, count: 3`

Task III

Populate a floating point array c[] with the following data:

`c[i] = a[i] + b[i]/10`^{d_i}, where d_i is the number of digits of the content of `b[i]`.

Input: `5`

Input: `233 87 13 6763 1597`

Output: `233.159700 87.676300 13.130000 6763.870000 1597.233000`

Evaluation

We treat elements of `a[]` to be coefficients of a polynomial

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$$

of degree $n-1$ where a_i is the data of `a[i]`. $a(x)$ can be evaluated for some value of x using $n-1$ multiplication.

Task IV

Write a function
`double eval(int a[], int n, double x)`,
that takes the array `a[]`, numnber data `n` and a value of `x` as parameters. It evaluates the polynomial using $n - 1$ multiplications.

Task IV

Read a value of `x` in `main()`. Call `eval(a,n,x)` and print the return value.
Input: `3`
Input: `1 2 3` $[a(x) = 3x^2 + 2x + 1]$
Input: `2.0`
Output: `17.000000`

Submission by ftp

```
$ ftp 10.5.17.186
Connected to 10.5.17.186.
220---------- Welcome to Pure-FTPd ----
220-You are user number 1 of 50 allowed.
220-Local time is now 07:54. ....... 21.
220-IPv6 connections ........
220 ... disconnected .. inactivity.
Name (10.5.17.186:...): pds
```

Submission by ftp

```
331 User pds OK. Password required
Password: pds04
230-User pds has group access to:  pds
230 OK. Current restricted directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd assignment6
250 OK. Current directory is /assignment6
```

Submission by ftp

```
ftp> put D0606.c
local: D0606.c remote: D0606.c
200 PORT command successful
150 Connecting to port 47093
226-File successfully transferred
226 0.001 seconds .. 39.00 Kbytes ..
27 bytes sent in 0.00 secs (1098.6 kB/s)
ftp> bye
21-Goodbye. .....
221 Logout.
$
```

Real Root of a Function

If $f(x)$ is a real valued function and x_n is a value close to a real root(zero) of $f(x)$, then a better approximation of the root is $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. This is known as Newton-Raphson method of finding real root. It starts with a '*reasonable*' initial guess of a point x_0 close to root and iterates to get better values.

Assignment VII

Write a C program that will read a floating-point number n and will call the function `double cbrt(double n)` that will use Newton-Raphson method to find the $\sqrt[3]{n}$. The function `main()` reads n, calls `double cbrt(double n)` to compute $\sqrt[3]{n}$, and prints the result. Assume suitable percentage error e.g. 0.0001%. **[Marks:** 5]

Assignment VII Output

```
cbrt(1.000000) = 1.000000
cbrt(-1.000000) = -1.000000
cbrt(64.000000) = 4.000000
cbrt(-64.000000) = -4.000000
cbrt(100000000.000000) = 464.158883
cbrt(-100000000.000000) = -464.158883
cbrt(0.000100) = 0.046416
cbrt(-0.000100) = -0.046416
```