مقدمه

روش نیوتن (یا نیوتن–رافسون)[1] یک روش عددی برای به‌دست آوردن ریشه‌ها (یا صفرها)ی یک تابع است. در این تمرین شما باید کتابخانه‌ای تولید کنید که امکان استفاده از این روش را برای هر تابع ریاضی فراهم نماید.

برای این کار برنامه‌ی شما باید به نحوی نوشته‌شود که تابع main() به شکل زیر به درستی ترجمه شده و با برنامه‌ی شما لینک شود.

```
#include <iostream>
#include "function.h"
#include "solver.h"
#include "polynomial.h"

using namespace std;

int main() {
    vector<double> v;
    v.push_back(1);
    v.push_back(2);
    v.push_back(3);
    v.push_back(4);
    Function* func = new Polynomial(v);
    cout << "One of the roots of the function: "
        << func->toString() << " is: "
        << findFunctionRoot(func, 100, 10) << endl;
    delete func;
    return 0;
}
```

بنا بر این لازم است که شما تابع findFunctionRoot() و کلاسهای Polynomial و Function را تعریف نموده و سازنده‌ها و توابع عضوی که در کد بالا فراخوانی شده‌اند را برای این کلاس‌ها تعریف کنید. تابع findFunctionRoot() دارای امضای زیر است:

```
double findFunctionRoot(Function* func, int iterations, double initialX);
```

آرگومان‌های این تابع به ترتیب مشخص‌کننده‌ی تابع مورد نظر، تعداد تکرارهای الگوریتم نیوتن–رافسون و نقطه‌ی شروع این الگوریتم هستند. تابع مورد نظر به صورت اشاره‌گری به یک شیء از نوع Function (یا فرزندان آن) مشخص می‌شود.

کلاس Function کلاسی انتزاعی است که دارای توابع عضو زیر است:

```
virtual double value(double x) = 0;
virtual Function* derivitive() = 0;
virtual string toString() = 0;
```

تابع عضو اول باید به نحوی پیاده‌سازی شود که مقدار تابع مورد نظر را در نقطه‌ی x داده‌شده محاسبه کند و تابع عضو دوم هم مشتق تابع را به صورت یک تابع دیگر بر می‌گرداند. این دو تابع عضو هر دو در تابع findFunctionRoot() استفاده می‌شوند. تابع سوم باید تابع داده‌شده را به صورت یک رشته (که در نمایش تابع در کنسول استفاده می‌شود) نمایش دهد. بدیهی است که هر سه تابع باید در کلاس‌هایی که از این کلاس ارث می‌برند پیاده‌سازی شوند.

همان‌طور که از تابع main() داده‌شده قابل تشخیص است، کلاس Polynomial یک نمونه از این کلاس‌ها است که توابع چندجمله‌ای را پیاده‌سازی می‌کند. شما باید این کلاس را نیز در برنامه‌ی خود در نظر بگیرید. لازم است مقادیر ضرایب چندجمله‌ای به صورت یک vector<**double**> از طریق سازنده برای این کلاس مشخص شوند و توابعی که از کلاس Function به ارث رسیده‌اند به شکل صحیحی پیاده‌سازی شوند. **در پیاده‌سازی تابع عضو value() برای کلاس Polynomial باید از روش بازگشتی استفاده کنید.**

در نظر داشته‌باشید که برنامه‌ی شما نباید تنها محدود به توابع چندجمله‌ای باشد؛ بلکه امکان اضافه کردن توابع مختلف به آن وجود داشته‌باشد.

به عنوان مثال کلاسی با اعلان زیر نیز باید قابلیت لینک و اجرا شدن به کمک برنامه‌ی شما را داشته‌باشد (این کلاس به عنوان مثال آورده شده و نیازی به پیاده‌سازی آن نیست):

[1] Newton's method (Newton-Raphson method)

```cpp
#include "function.h"

// a.Sin(mx) + b.Sin(nx)
class Trigonometric: public Function {
public:
    Trigonometric(double a, double m, double b, double n);
    virtual Function* derivitive();
    virtual double value(double x);
    virtual string toString();
private:
    double a, m, b, n;
};
```

دقت داشته‌باشید که برنامه‌ی شما باید به صورت یک کتابخانه‌ی ایستا[۲] عمل کند به این صورت که با اجرای دستور make فایلی با عنوان libsolver.a تولید شود. ما برای بررسی برنامه‌ی شما یک فایل شامل تابع main و (احتمالاً) چند فایل شامل توابعی که خودمان تعریف می‌کنیم را تهیه می‌کنیم. دو هدرفایل function.h و solver.h و (احتمالاً) polynomial.h که توسط شما نوشته شده‌اند را در ابتدای این فایل‌ها **#include** می‌کنیم و بعد از ترجمه کتابخانه‌ی شما را به آنها لینک نموده و نتیجه‌ی اجرای آن را بررسی می‌کنیم.

نحوه‌ی تحویل

شما باید فایل‌های برنامه‌ی خود (فقط شامل فایل‌های h. و cpp. که هیچ‌یک دارای تابع main نباشند) را به همراه یک Makefile در یک پوشه با نام A8-SID قرار داده و آن را با فرمت zip آرشیو کنید. این فایل را (با نام A8-SID.zip) در سایت درس آپلود نمایید. SID پنج رقم آخر شماره‌ی دانشجویی شماست. مثلاً اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۲۱۲۳ است، نام فایل شما باید A8-92123.zip باشد.

بارم‌بندی

طراحی شیءگرا و پیاده‌سازی صحیح چندریختی	۳۰
پیاده‌سازی صحیح الگوریتم بازگشتی	۱۰
تقسیم‌بندی صحیح فایل‌ها	۱۰
نگارش Makefile استاندارد	۱۰
تولید کتابخانه به روش اساندارد	۱۰
اجرای صحیح برنامه با توابع مختلف	۳۰
مجموع	**۱۰۰ نمره**

دقت کنید

- برنامه‌ی شما باید تحت سیستم عامل لینوکس نوشته‌شده، و با استفاده از مترجم g++ قابل ترجمه باشد.
- شما باید تمام قابلیت‌های ذکرشده در صورت تمرین را به شکل کامل در طراحی خود پیاده‌سازی نمایید.
- به فرمت و نام فایل‌های خود دقت کنید. در صورتی که هر یک از موارد گفته‌شده رعایت نشود، نمره‌ی صفر برای شما در نظر گرفته می‌شود.
- در صورت کشف تقلب در کل و یا قسمتی از تمرین، برای هر دو طرف نمره‌ی ۱۰۰− منظور خواهد شد.

[۲] Static Library، برای یادگیری چگونگی انجام این کار از اینترنت استفاده کنید!