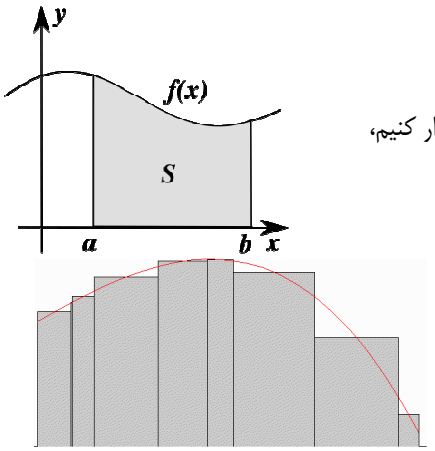
یادآوری: تمامی تمرینات و اطلاعات مربوط به تحویل آن‌ها بصورت هفتگی در سایت درس قرار داده می‌شوند:

http://ele.aut.ac.ir/~btaheri/cpp/

توجه ۲: هنگام ارسال برنامه،کافیست فقط فایل cpp. رو ارسال کنید. (به هیچ وجه فایل exe. که فایل اجرایی است که توسط کامپایلر ساخته شده است را نفرستید!)

سوالات برنامه نویسی

۱- هدف: تقریب تابع انتگرال:



فرض کنیم می خواهیم مقدار مساحت زیر نمودار یک تابع را حساب کنیم: $S = \int_a^b f(x)dx$

در صورتی که بازه ی $[a,b]$ را به به بازه های کوچک تر بصورت $a = x_0 < x_1 < ... < x_n = b$ افزار کنیم، میتوانیم انتگرال فوق را به صورت زیر تقریب بزنیم:

$$S \approx \sum_{i=1}^{n-1} f(t_i)\left(x_{i+1} - x_i\right)$$

که در آن $\left(x_{i+1} - x_i\right)$ طول بازه و $f(t_i)$ ارتفاع بازه است. t_i نقطه ی متناظر با هر بازه ی $[x_i, x_{i+1}]$ که می تواند طبق قرارداد، هر نقطه ای از آن باشد. انتخاب های مهم برای می توانند مقادیر ذیل باشند:

۱. انتهای بازه: $t_i = x_{i+1}$
۲. ابتدای بازه: $t_i = x_i$
۳. وسط بازه: $t_i = \frac{x_i + x_{i+1}}{2}$

سوال: برنامه ای بنویسید که انتگرال تابع $\sin(x) + \log(x^2) + x^7$ را بصورت تقریبی، با استفاده از روش افراز بازه به نواحی با عرض یکسان در بازه ی $[1,2]$ بدست آورد.

اطلاعات: برنامه باید مقدار N که تعداد افراز مورد نظر است و Type که نوع انتخاب t_i به هرکدام از سه روش(انتهای بازه، ابتدای بازه و میان بازه) را از ورودی بگیرد و مقدار انتگرال را در خروجی نمایش دهد.

اطلاعات۲: ضمیمه ی برنامه یک فایل word نیز قرار دهید که دارای محتویات زیر باشد:

تعداد افراز N	$t_i = x_{i+1}$	$t_i = x_i$	$t_i = \frac{x_i + x_{i+1}}{2}$
5			
20			
200			

هر خانه ی خالی جدول فوق باید دارای مقدار تقریبی انتگرال توسط برنامه ی شما باشد. در واقع با جدول فوق، عملکرد برنامه را در تقریب انتگرال به ازای انتخاب های مختلف بررسی می کنیم.

در ادامه ی فایل word به سوالات زیر نیز پاسخ دهید:

۱. مقدار دقیق انتگرال را بنویسید.
۲. با توجه به مقدار جدول، افزایش N چه تاثیری در دقت دارد.
۳. با توجه به مقادیر جدول، انتخاب های مختلف t_i چه تاثیری در دقت دارد.

اطلاعات۳: می توانید از دستور `setprecision()` که در کتابخانه ی `iomanip` تعریف شده است، استفاده کرده و دقت نمایش را افزایش دهید.

اطلاعات۴: موقع ارسال تمرین فایل cpp. را همراه با فایل word بفرستید.
اطلاعات۵: برنامه بسیار آسان است!

۲- برنامه ای بنویسید که سری فیبوناچی را تا N جمله حساب کرده و در خروجی نمایش دهد.
یادآوری: سری فیبوناچی: $a_{n+1} = a_n + a_{n-1}, a_1 = a_2 = 1$
توجه: از توابع استفاده بکنید و برنامه را بصورت بازگشتی (recursive) پیاده سازی کنید.
توجه: از حلقه ها استفاده نکنید!
راهنمایی: نمونه ای از استفاده از توابع به صورت بازگشتی در انتهای handout شماره ی ۸ برای محاسبه ی فاکتوریل قرار دارد.
نمونه:

INPUT:
5

OUTPUT:
1 1 2 3 5

همچنین: یک فایل word درست کنید و به سوالات زیر جواب بدید:

۱. در حالتی که از متغیر int برای ذخیره سازی مقادیر دنباله استفاده کرده اید، حداکثر مقداری از دنباله که برنامه می تواند حساب کند را تعیین کنید. توضیح دهید چرا نمیتوان بالاتر از آن را حساب کرد.
۲. برای افزایش محدوده ی محاسبات از متغیر double استفاده کنید. در این حالت نیز حداکثر مقداری از دنباله که برنامه می تواند حساب کند را تعیین کنید. توضیح دهید چرا نمیتوان بالاتر از آن را حساب کرد.
۳. به یاد دارید در تمرین سری قبل همین سوال را با حلقه ها پیاده سازی کرده بودید. با استفاده از CodeBlocks که زمان اجرای برنامه را به شما می دهد، جدول زیر را کامل کنید:

تعداد جملات N	زمان اجرا با استفاده از روش حلقه ها (تمرین سری قبل)	زمان اجرا با استفاده از روش بازگشتی
5		
15		
30		

توجه: اگر زمان اجرای برنامه متغیر است، برنامه را چندین بار اجرا کنید، زمان ها را یادداشت کرده و از آنها میانگین بگیرد.

۴. **با توجه به جدول این سوال پاسخ دهید:** کدامیک از پیاده سازی این سوال(بازگشتی یا پیاده سازی مستقیم با استفاده از حلقه ها) سریع تر عمل می کنند؟ چرا؟

۳- **تفاوت بین call by reference و call by value:** سوالات ۳و ۴و ۵ فصل ۵ کتاب (توابع) رو حل کنید!
اطلاعات۱: لازم نیست جوابتون رو به ما بدید! فقط **صادق باشید** و حتما این تمرینات رو حل کنید تا به تفاوت دو شیوه ی فراخوانی فوق پی ببرید.
اطلاعات۲: اگه متوجه نشدید در تمرینات handout در کلاس از تدریسیار کمک بگیرید.
اطلاعات۳: لازم نیست جوابتون رو به ما بدید!! (محض تاکید!)

۴- **هدف:** پیدا کردن اعداد اول با یک روش دیگر؛ پازل اراتوتسن[1]:

فرض کنید N عددی باشد که قصل داریم اعداد اول کوچکتر از آن را بدست آوریم. برای مثال N=10. ابتدا یک آرایه ی ده خانه ای تعریف مـی کنـیم. در خانه ی اول عدد 2 قرار داده و بقیه را به صورت صعودی(با اختلاف یک واحد) پر می کنیم. لذا باید داشته باشیم:

2	3	4	5	6	7	8	9	10

المان اول را در نظر بگیرد و در طول آرایه، بعد از 2، حرکت کنید و تمام خانه هایی که قابل تفسیم بر 2 هستند را حذف کنید. (به عنوان قرار داد چـون مـی دانیم در محدوده ی اعداد طبیعی بحث می کنیم، مقدار صفر را معادل حذف یک خانه در نظر می گیریم.) حال داریم:

2	3	0	5	0	7	0	9	0

حال سراغ عدد غیر صفر بعدی یعنی 3 می رویم و دوباره عمل فوق را دوباره انجام می دهیم. حال داریم:

2	3	0	5	0	7	0	0	0

این کار را تا انتهای آرایه ادامه می دهیم تا اینکه همه ی اعداد اول باقی بمانند:

2	3	0	5	0	7	0	0	0

سوال: برنامه ای بنویسید که با استفاده از روش فوق و آرایه ها(یا وکتور ها)، اعداد اول را چاپ کند.

۵- **هدف:** آرایه به عنوان ورودی یک تابع:

برنامه ی سوال ۲ را در نظر بگیرد. هدف این است که تابعی بنویسید که الگوریتم یافتن اعداد اول با استفاده ازآرایه را در داخل یک تابع انجام دهد. مراحل کار باید به اینصورت باشد:

۱. یک آرایه به نام []primeArray در تابع ()main ایجاد شود.
۲. آرایه ی []primeArray و اندازه ی آن به ورودی تابع ()primeArrayFunc که دارای خروجی void است داده شود.
۳. بعد از اجرای برنامه و محاسبات اعداد اول در داخل تابع، اعداد اول در تابع ()main و از روی آرایه ی []primeArray چاپ شود.

یادآوری: ارجاع آرایه به تابع بصورت Call By Reference انجام میشود!

۶- **هدف:** تقریب عدد π با استفاده از تکرار های زیاد!

همانطور که می‌دانید، پرتاب یک تیر به سمت نشانه، در تکرار زیاد، عملی تصادفی محسوب می‌شود. حال دایره‌ای را که در داخل مربعی محاط شـده اسـت، تصوّر کنید. به راحتی می‌توان نشان داد که نسبت مساحت دایره به مربع برابر است با $\frac{\pi}{4}$. در واقع در اینجا می‌خواهیم روشـی را ارائـه کنـیم کـه بتـوانیم بـا پرتاپ‌های متوالی، مقدار عدد پی را با استفاده از نسبت مساحت ها، حساب کنیم!

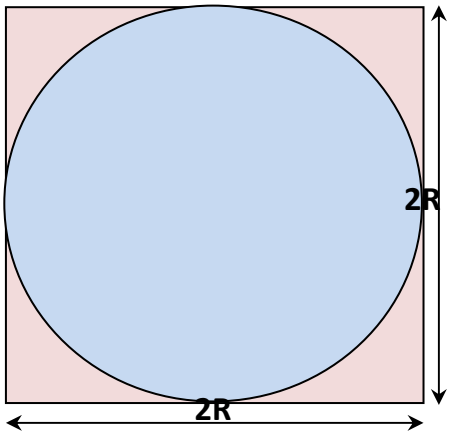
- به صورت تصادفی شروع به پرتاب دارت به سمت مربع می‌کنیم!
- دارت‌هایی که به خارج از مربع برخورد می کنند را نادیده می گیریم!
- دارت‌هایی را که به داخل دایره برخورد می کنند، می‌شماریم!

اگر N_1 تعداد برخورد دارت‌هایی باشد که به دایره برخورد کرده است و N_2 تعداد دارت‌هایی باشد که به مربع برخورد کرده است، چون احتمال برخـورد دارت به مربع در پرتاپ های زیاد برابر با نسبت مساحت‌های آنهاست، داریم:

$$S_1 = \pi R^2$$

$$S_2 = 4R^2 \quad \Longrightarrow \quad P = \frac{S_1}{S_2} = \frac{\pi}{4} \quad \Longrightarrow \quad \pi = 4P = 4\frac{N_1}{N_2}$$

[1] **Eratosten's Riddle**



سوال: با استفاده از اطلاعات داده شده، برنامه ای بنویسید که عدد π را به صورت تصادفی محاسبه کند.

اطلاعات۱: تابع `rand()` در کتابخانه ی `stdlib` تعریف می شود. خروجی تابع یک عدد صحیح تصادفی بین صفر تا عدد ثابت از پیش تعریف شده ی `RAND_MAX-1` است. برای مثال اگر بخواهیم یک عدد تصادفی حقیقی بین صفر و یک تولید کنیم:

```
float  a = rand()/RAND_MAX;
```

یا برای تولید اعداد تصادفی بین 1.2 و 2.5 کافیست:

```
float  a = 1.2 + 1.3 * rand() / RAND_MAX;
```

یا برای تولید اعدد صحیح تصادفی از مجموعه ی $\{0,1\}$ داریم:

```
int  a = round( rand() / RAND_MAX );
```

یادآوری: تابع `round()` عدد را گرد می کند. تابع `floor()` عدد را به پایین گرد می کند. تابع `ceil()` عدد را به بالا گرفت می کند!

اگر تعداد عدد تصادفی را به اینصورت چاپ کنید مشاهده خواهید کرد که ترتیب اعداد تصادفی در اجرا های مختلف یکسان است. اما چرا؟ باید گفت تابع تولید اعداد تصادفی هرچه باشد، دارای یک الگوریتم مشخص است و به ازای اجرا های مختلف انتظار اینکه اعداد تصادفی با ترتیب یکسان بدهد قابل پیش بینی است. لذا لفظ random یاد تصادفی به خروجی این توابع کاملا تقریبی است. چرا که خروجی این توابع در اصل pseudo-random (شبه تصادفی) اند. طراحان C++ برای اینکه خروجی تابع `rand()` تصادفی تر شود، راه چاره ای در نظر گرفته اند؛ می توان ارتباطی بین تابع `rand()` و ساعت رایانه ایجاد کرد تا خروجی آن با گذر زمان دائما عوض شود!

ادامه اطلاعات ۱: تابع ()time: تابع زمان گذشته شده از نصفه شب ژانویه ی ۱۹۸۰ تاکنون را به عنوان خروجی می دهد!! برای استفاده از این تابع آن را باید به صورت `time(NULL)` بکار برد(NULL یک اشاره گر کمکی برای اشاره به نقطه ای غیر مشخص است!! هنور مباحث اشاره گر ها را نخوانده ایم). برای مثال وارد کنید:

```
cout << time(NULL) << endl;
```

مشاهده خواهید کرد که حاصل یک عدد صحیح است که دائما در حال زیاد شدن است! (البته در اجرا های مختلف!) این تابع در کتابخانه ی `time.h` تعریف میشود.

ادامه اطلاعات ۱: تابع ()srand: این تابع همان تابع پدر سوخته ای است که `rand()` را به `time()` ربط می دهد!! کافیست هر وقت می خواهید از تابع `rand()` در برنامه ها استفاده کنید، در ابتدای اجرای برنامه وارد کنید:

```
srand(time(NULL));
```

و سپس از تابع `rand()` استفاده کنید.

اطلاعات۵: برنامه بسیار آسان است!