- تنها فایل کد برنامه (.cpp) مربوط به هر تمرین را ارسال نمایید. هر فایل را مطابق نام تعیین شده برای همان سوال نام‌گذاری کنید. *توجه کنید در انتهای نام هر فایل باید سه رقم انتهایی شماره دانشجویی خود را نیز بنویسید.*

در نهایت، مجموعه کدهای برنامه‌های هر سری از تمارین را در یک پوشه zip نمایید. نام فایل zip ارسالی باید مطابق دستور زیر باشد:

HW#-StudentID.zip

به عنوان مثال اگر شماره دانشجویی شما ۹۳۰۰۲۲۲۱۲۳ است، نام فایل ارسالی شما برای تمرین اول HW1-9300222123.zip خواهد بود.

فایل نهایی را به آدرس ایمیل coursetoroghi+C93HW@gmail.com ارسال نمایید. موضوع ایمیل (subject) همان نام فایل (در مثال بالا HW1-9300222123) باید باشد.

در صورت عدم رعایت هر کدام از موارد بالا بین ۵ تا ۳۰ درصد از نمره شما کاسته خواهد شد.

- مهلت ارسال برنامه‌ها ساعت ۲۴ شب تاریخ اعلام شده است. تا ۶ ساعت تاخیر ۱۵ درصد از نمره شما کاسته می‌شود. تا ۲۴ ساعت تاخیر ۳۰ درصد از نمره شما کاسته می‌شود. تا ۴۸ ساعت تاخیر ۵۰ درصد از نمره شما کاسته می‌شود. بعد از ۴۸ ساعت هیچ نمره‌ای به تمارین ارسالی تعلق نخواهد گرفت.
- برنامه‌های نوشته شده باید قابل اجرا باشند و دقیقا خروجی مورد نظر را بدهند. به عنوان نمونه اگر خروجی مورد انتظار از یک برنامه خط زیر باشد:

1,2,3

و برنامه شما خروجی زیر را بدهد:

1 2 3

هیچ نمره‌ای به شما تعلق نمی‌گیرد. لطفا در این زمینه دقت کافی را مبذول فرمایید.

- مشورت آزاد است، ولی به هیچ‌وجه تقلب نکنید. تقلب‌ها به سادگی قابل تشخیص هستند. اولین باری که از یک دانشجو تقلب گرفته شود، علاوه بر از دست دادن نمره آن تمرین، یک نمره منفی نیز به وی داده می‌شود. دومین باری که از یک دانشجو تقلب گرفته شود، نمره کل تمارین (حدود ۵ نمره از ۲۰ نمره) را از دست خواهد داد. سومین باری که از یک دانشجو تقلب گرفته شود، در این درس نمره قبولی را نخواهد گرفت. کپی کردن کد از اینترنت تقلب محسوب می‌شود.
- نمره این تمرین از ۱۰۰ است (این تمرین ۴۰ نمره اضافه دارد).

۱) (۳۰ نمره) (Start-123.cpp)[۱] برنامه‌ای بنویسید که سه تابع زیر را دارا باشد. در تابع main برنامه لازم نیست کد خاصی بزنید، این تابع توسط مصحح سوالات پر می‌شود:

الف) تابعی که در ورودی یک آرایه از اعداد صحیح (int []) و اندازه آن را دریافت کند و خروجی آن اندیس ماکزیمم آرایه باشد (دقت کنید اندیس از صفر شروع می‌شود).

ورودی نمونه: {3, 34, 23, 78, -34, 24, 10}, 7

خروجی نمونه: 3

ب) تابعی بنویسید که در ورودی سه آرایه و اندازه‌ی آنها را دریافت کند (فرض کنید آرایه‌ها هم اندازه هستند). این تابع باید هر اندیس این آرایه‌ها را مرتب کند به گونه‌ای در اندیس i-ام از آرایه اول کوچک‌ترین مقدار آن اندیس در هر سه آرایه قرار داشته باشد و در اندیس i-ام از آرایه سوم بزرگ‌ترین مقدار آن اندیس در هر سه آرایه قرار داشته باشد.

ورودی نمونه: {12, 63, 140, 28, 9}, {0, -4, 42, 11, 20}, {30, 34, 23, 78, -34}, 5

وضعیت آرایه‌ها پس از اتمام کار تابع: {30, 63, 140, 78, 20}, {12, 34, 42, 28, 9}, {0, -4, 23, 11, -34}

ج) تابعی بنویسید که در ورودی دو آرایه و اندازه‌ی آنها را دریافت کند (فرض کنید آرایه‌ها هم اندازه هستند). این تابع باید این دو آرایه را با هم جابجا نماید.

ورودی نمونه: {32, 63, 10, 17, -1}, {13, 90, 49, -4, 20}, 5

وضعیت آرایه‌ها پس از اتمام کار تابع: {13, 90, 49, -4, 20}, {32, 63, 10, 17, -1}

۲) (۳۰ نمره) (Khayyam-123.cpp) برنامه‌ای بنویسید که از کاربر عدد n را دریافت کند و سپس بسط دو جمله‌ای خیام-نیوتون[۲] به توان n را نمایش دهد. به عنوان نمونه به ازای ورودی ۵، خروجی برنامه شما عبارت است از:

$$a^5 + 5a^4b^1 + 10a^3b^2 + 10a^2b^3 + 5a^1b^4 + b^5$$

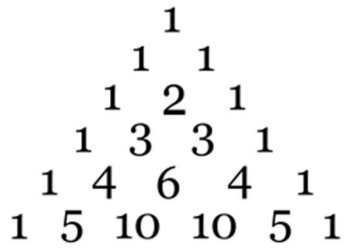
برای نوشتن این برنامه دو تابع با مشخصات زیر بنویسید. وظیفه تابع اول یافتن ضرایب بسط دو جمله‌ای به کمک مثلث خیام-پاسکال[۳] است. این تابع یک عدد (n) و یک اشاره‌گر به عدد (int*) دریافت می‌کند و در این اشاره‌گر یک آرایه به اندازه n+1 می‌ریزد، که این آرایه همان سطر n+1-ام مثلث خیام-پاسکال است. وظیفه تابع دوم نمایش جواب است. این تابع نیز یک عدد (n) و یک اشاره‌گر به عدد (پر شده توسط تابع قبلی) دریافت می‌کند و بسط دوجمله‌ای توان n را به کمک اعداد موجود در آرایه‌ای که اشاره‌گر ورودی به آن اشاره می‌کند، نمایش می‌دهد.

---

[۱] سه رقم آخر در نام فایل، سه رقم انتهایی شماره دانشجویی شما است.

[۲] برابر است با $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$، برای آشنایی بیشتر می‌توانید به صفحه ویکی‌پدیا بسط دوجمله‌ای مراجعه کنید.

[۳] برای آشنایی با این مثلث می‌توانید به صفحه ویکی‌پدیا مثلث-خیام پاسکال مراجعه کنید.



شکل ۱- ۶ سطر اول مثلث خیام-پاسکال

توجه داشته باشید، در صورتی که از روش مثلث خیام-پاسکال و یا پیاده‌سازی توابع به صورت تشریح شده در بالا استفاده نکنید، هیچ نمره‌ای به شما تعلق نمی‌گیرد.

۳) (۴۰ نمره) (Power-123.cpp) برنامه‌ای بنویسید که در ورودی عدد n را دریافت کند و در خروجی $(2)^{-n}$ را با دقت کامل چاپ نماید. دقت کنید که برای این کار نمی‌توانید از متغیرهای double، long، و ... استفاده کنید و برای محاسبه و نمایش این اعداد باید از یک آرایه از اعداد استفاده کنید (هر خانه از آرایه برای ذخیره سازی یک رقم است). حل این سوال با استفاده از کتابخانه‌های موجود هیچ نمره‌ای برای شما در پی نخواهد داشت.

ورودی نمونه	خروجی نمونه
۳	۰,۱۲۵
۱۲	۰,۰۰۰۲۴۴۱۴۰۶۲۵
۷۰	۰,۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۸۴۷۰۳۲۹۴۷۲۵۴۳۰۰۳۳۹۰۶۸۳۲۲۵۰۰۶۷۹۶۴۱۹۶۲۰۵۱۳۹۱۶۰۱۵۶۲۵

۴) (۴۰ نمره) (Queens-123.cpp) هدف پیدا کردن راه‌حلی برای چیدن n وزیر در صفحه شطرنج n*n است. به نحوی که هیچ دو وزیری یکدیگر را تهدید نکنند. برای این کار از روش کمترین مقدار باقی‌مانده استفاده می‌کنیم. به این ترتیب که:

۱) هر ستون را به عنوان یک عنصر در نظر می‌گیریم که باید یک وزیر را در یکی از سطرهایش قرار دهیم.
۲) وزیر اول را در سطر اول از ستون اول قرار می‌دهیم.
۳) حال برای هر ستون تعداد خانه‌هایی از آن را که تهدید نمی‌شوند، می‌شماریم.
۴) برای مرحله بعد، ستونی را که کمترین خانه‌ی امن را دارد انتخاب می‌کنیم و یک وزیر را در اولین سطر امن آن قرار می‌دهیم (اگر دو ستون تعداد خانه‌ی امن کمینه و مساوی داشته باشند، ستون با اندیس کمتر را انتخاب می‌کنیم).
۵) مراحل ۳ و ۴ را تکرار می‌کنیم تا به یکی از دو حالت زیر برسیم:

- در هر ستون یک وزیر در خانه‌های امن قرار می‌گیرد که در این صورت موفق شده‌ایم و مختصات وزیران را چاپ می‌کنیم.
- با وجود تمام نشدن وزیران، دیگر هیچ خانه‌ای امنی برای بقیه ستون‌ها باقی نمانده است. بنابراین آخرین وزیری را که قرار داده بودیم به خانه‌ی امن بعدی در همان ستون منتقل می‌کنیم. اگر آن ستون دیگر خانه‌ی امنی ندارد، پس باید به ستون قبلی که وزیری را در آن قرار داده بودیم، برویم و همین عملیات را تکرار کنیم.

برای این مسئله تضمین می‌شوند که حتماً جوابی به روش فوق پیدا می‌شود. پیاده‌سازی این الگوریتم به روش بازگشتی بسیار ساده است. به این ترتیب که در هر مرحله، یک صفحه شطرنج با تعدادی وزیر روی آن و تعداد خانه‌های امن هر ستون را داریم. لذا ستون موردنظر را انتخاب می‌کنیم و به ازای هر یک از سطرهای امن آن، به ترتیب وزیری در آن قرار می‌دهیم، خانه‌های امن باقیمانده را بررسی می‌کنیم و دوباره تابع بازگشتی را صدا می‌زنیم. اگر به جواب رسیدیم، جواب را چاپ می‌کنیم و برنامه را پایان می‌دهیم. در غیر این صورت تابع بازگشتی با دادن خروجی false می‌گوید که سطرهای امن بعدی نیز باید بررسی شوند.

در ورودی عدد n را به عنوان تعداد سطرها و ستون‌های جدول بخوانید. در خروجی n عدد پشت سر هم بنویسید که هر کدام نشان‌دهنده‌ی محل وزیر در یکی از ستون‌ها (به ترتیب از چپ به راست) است. خانه‌ی اول هر ستون را شماره یک و خانه‌ی آخر هر ستون را با شماره‌ی n مشخص کنید.

به عنوان نمونه، برای n=8 داریم:

Q1	*	*	*	*	*	*	*
*	*						
*		*					
*			*				
*				*			
*					*		
*						*	
*							*
0	6	6	6	6	6	6	6

Q1	*	*	*	*	*	*	*
*	*	*					
*	Q2	*	*	*	*	*	*
*	*	*	*				
*	*		*	*			
*	*			*	*		
*	*				*	*	
*	*					*	*
0	0	4	4	4	4	4	5

Q1	*	*	*	*	*	*	*
*	*	*	*		*		
*	Q2	*	*	*	*	*	*
*	*	*	*				
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*		
*	*	*		*	*	*	
*	*	*			*	*	*
0	0	0	3	3	1	3	4

Q1	*	*	*	*	*	*	*
*	*	*	*		*		*
*	Q2	*	*	*	*	*	*
*	*	*	*	Q4	*	*	
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*		*
*	*	*		*	*	*	
*	*	*			*	*	*
0	0	0	2	2	0	2	1

Q1	*	*	*	*	*	*	*
*	*	*	*		*		*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	Q4	*	*
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*	*	Q5
*	*	*			*	*	*
0	0	0	1	2	0	1	0

Q1	*	*	*	*	*	*	*
*	*	*	*		*		*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	Q4	*	*
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	Q5
*	*	*	Q6	*	*	*	*
0	0	0	0	1	0	1	0

Q1	*	*	*	*	*	*	*
*	*	*	*	Q7	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	Q4	*	*
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	Q5
*	*	*	Q6	*	*	*	*
0	0	0	0	0	0	0	0

در نهایت در این حالت به جواب نرسیدیم. بنابراین آخرین وزیر (Q_7) را به خانه‌ی ممکن بعدی منتقل می‌کنیم. چون هیچ خانه امن دیگری در آن ستون نیست، به سراغ ستون قبل از آن می‌رویم. چون وزیر ششم، پنجم، و چهارم نیز خانه‌ی امن دیگری برای جابجایی ندارند، به سراغ وزیر سوم می‌رویم و آن را به خانه‌ی بعدی منتقل می‌کنیم:

Q1	*	*	*	*	*	*	*
*	*	*				*	
*	Q2	*	*	*	*	*	*
*	*	*	*	*			
*	*	*	*				
*	*	Q3	*	*	*	*	*
*	*	*	*		*	*	
*	*	*		*		*	*
0	0	0	2	3	4	2	4

Q1	*	*	*	*	*	*	*
*	*	*	Q4	*	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	*		
*	*	*	*			*	
*	*	Q3	*	*	*	*	*
*	*	*	*		*	*	
*	*	*	*	*		*	*
0	0	0	0	2	2	1	3

Q1	*	*	*	*	*	*	*
*	*	*	Q4	*	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	*	Q5	*
*	*	*	*		*	*	*
*	*	Q3	*	*	*	*	*
*	*	*	*		*	*	
*	*	*	*	*		*	*
0	0	0	0	2	1	0	1

Q1	*	*	*	*	*	*	*
*	*	*	Q4	*	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	Q5	*	
*	*	*	*	Q7	*	*	
*	*	Q3	*	*	*	*	
*	*	*	*	*	*	*	Q8
*	*	*	*	*	Q6	*	*
0	0	0	0	0	0	0	0

Q1	*	*	*	*	*	*	*
*	*	*	Q4	*	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	*	Q5	*
*	*	*	*		*	*	*
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*	*	
*	*	*	*	*	Q6	*	*
0	0	0	0	1	0	0	1

Q1	*	*	*	*	*	*	*
*	*	*	Q4	*	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	Q5	*	
*	*	*	*	Q7	*	*	
*	*	Q3	*	*	*	*	
*	*	*	*	*	*	*	
*	*	*	*	*	Q6	*	*
0	0	0	0	0	0	0	1

Q1	*	*	*	*	*	*	*
*	*	*	Q4	*	*	*	*
*	Q2	*	*	*	*	*	*
*	*	*	*	*	*	Q5	*
*	*	*	*	Q7	*	*	*
*	*	Q3	*	*	*	*	*
*	*	*	*	*	*	*	Q8
*	*	*	*	*	Q6	*	*
0	0	0	0	0	0	0	0

در این حالت موفق شدیم به جواب برسیم.

ورودی نمونه	خروجی نمونه
۸	۱۳۶۲۵۸۴۷

موفق باشید