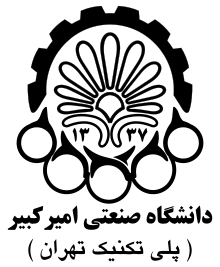
**بسم الله الرحمن الرحیم**



**گزارش تمرین سری پنجم**

**برنامه سازی پیشرفته**

**سیده معصومه سجادی**

**ش.د:۹۳۲۳۰۸۷**

خرداد 1398

**سوال1:**

آ)جهت حل این سوال ابتدا تابع fill را برای پر کردن وکتورمان بدون استفاده از حلقه تعریف میکنیم و همچنین برای نمایش خروجی نیز تابع print\_vector را مینویسیم.

ب)در بخش بعدی وکتور ها را تعریف میکنیم و مانند خواسته سوال آنها را پر و نمایش میدهیم و با دستورزیر وکتور یک را به انتهای وکتور دوم اضافه میکنیم :

vec2.insert(vec2.end(), vec1.begin(), vec1.end());

ج)وکتوری به نام odd\_vector تعریف میکنیم و مقادیر فرد وکتور اول را در آن میریزیم.بدین منظور از کد زیر بهتره میبریم:

std::vector<int> odd\_vec(vec1.size());

auto end\_odd\_vec = std::copy\_if(begin(vec1), end(vec1), begin(odd\_vec),

[](int n) { return n % 2 == 1; });

odd\_vec.erase(end\_odd\_vec, end(odd\_vec));

د)برای این بخش reverse\_vector را تعریف کرده و وکتور را از اخر به اول در آن مینویسیم:

revers\_vec = vec1;

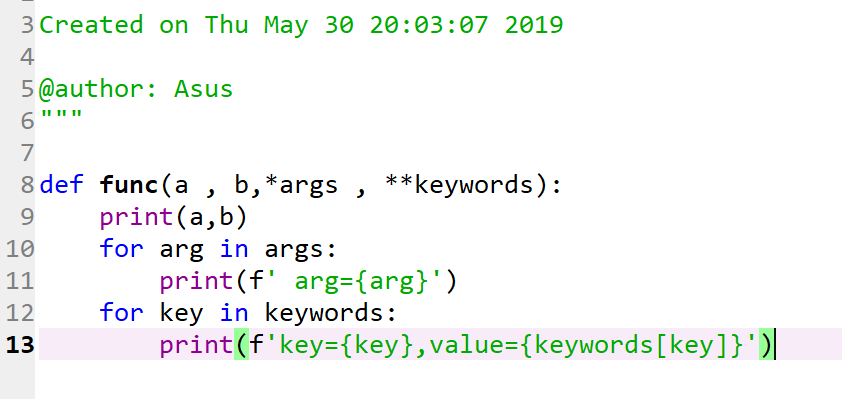
std::reverse(revers\_vec.begin(), revers\_vec.end());

ه) برای پردازش موازی از دستورات std::execution::par استفاده میشود و باید دقت شود که کتابخانه ی #include <execution> را بیافزاییم و همینطور باید ورژن برنامه visual studio نیز حتما روی c++2017 تنظیم شود تا برنامه به درستی اجرا شود.

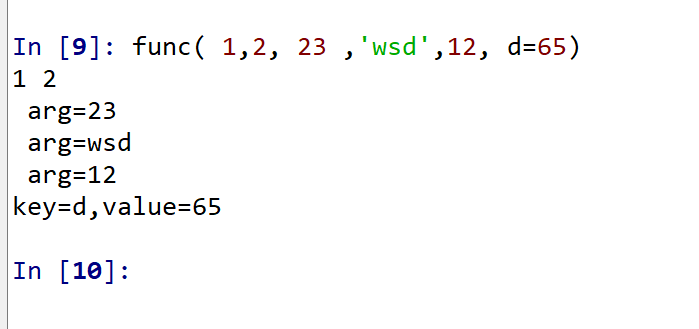
کامنت های مربوطه نیز در کد موجود میباشد.

**سوال 2:**

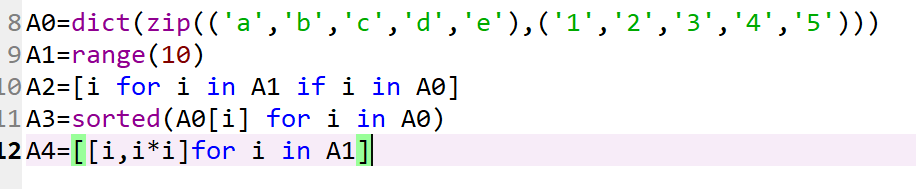
در پایتون با گذاشتن یک ستاره پشت متغیر نشان میدهیم که منظور ما tuple است و هر گاه دو ستاره بگذاریم نشان دهنده ی دیکشنری می باشد.به عنوان مثال در زیر مشاهده میکنیم که ورودی اول و دوم فانکشن را در حالت عادی نشان داده ایم.ورودی های بعدی را هر چندتا که باشند در arg که tuple است میریزیم و اگر ورودی به صورت دیکشنری باشد آن را به صورت دیکشنری نمایش میدهد:



با تست مشاهده میکنیم:



**سوال 3:**



خروجی عبارت اول یک دیکنشری به ما میدهد که به صورت زیر است و هر عدد یک حرف را نمایش میدهد.

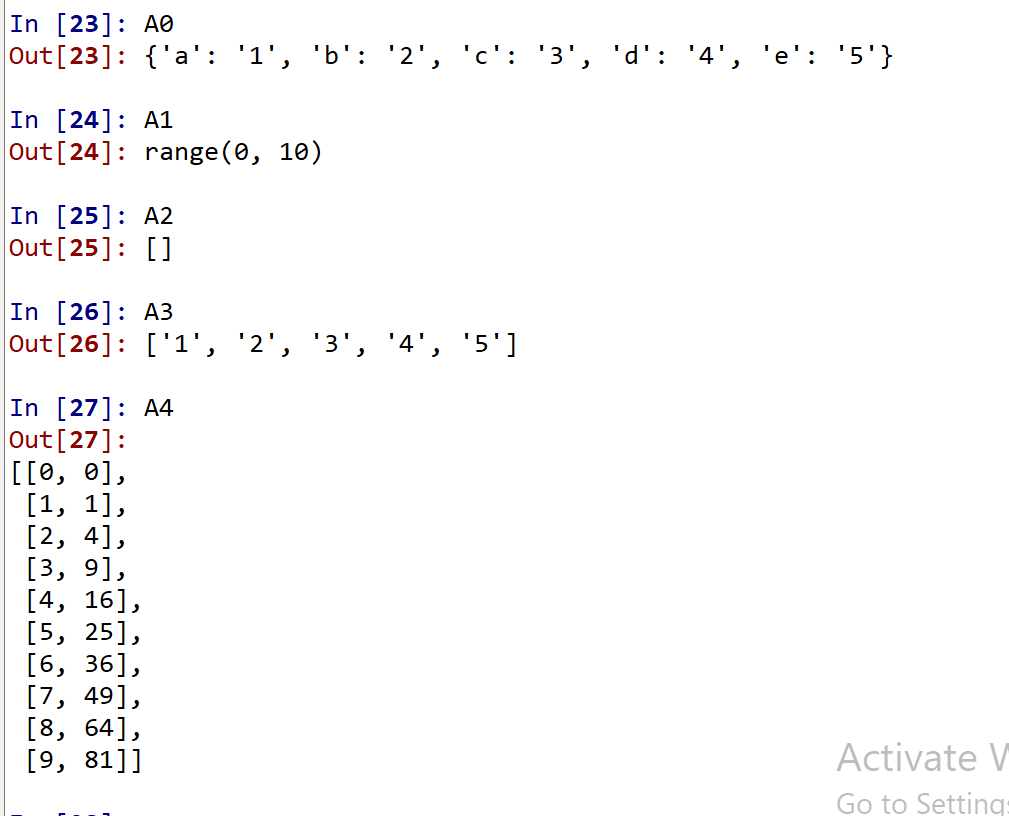
خروجی A1 یک رشته از 0 تا 9 را به ما میدهد.

A2 عبارات مشترک بین A1 و A0 است.

مقدار 3Aسورت شده ی A0 است که بر اساس المان های A0 است.

و A4 نشان هر درایه را به همراه توان دوم آن نشان میدهد.

برای چک کردن این پاسخ ها عبارات را در پایتون وارد میکنیم:



**ب) متاسفانه** متوجه منظور سوال نشدم.

**سوال4:**

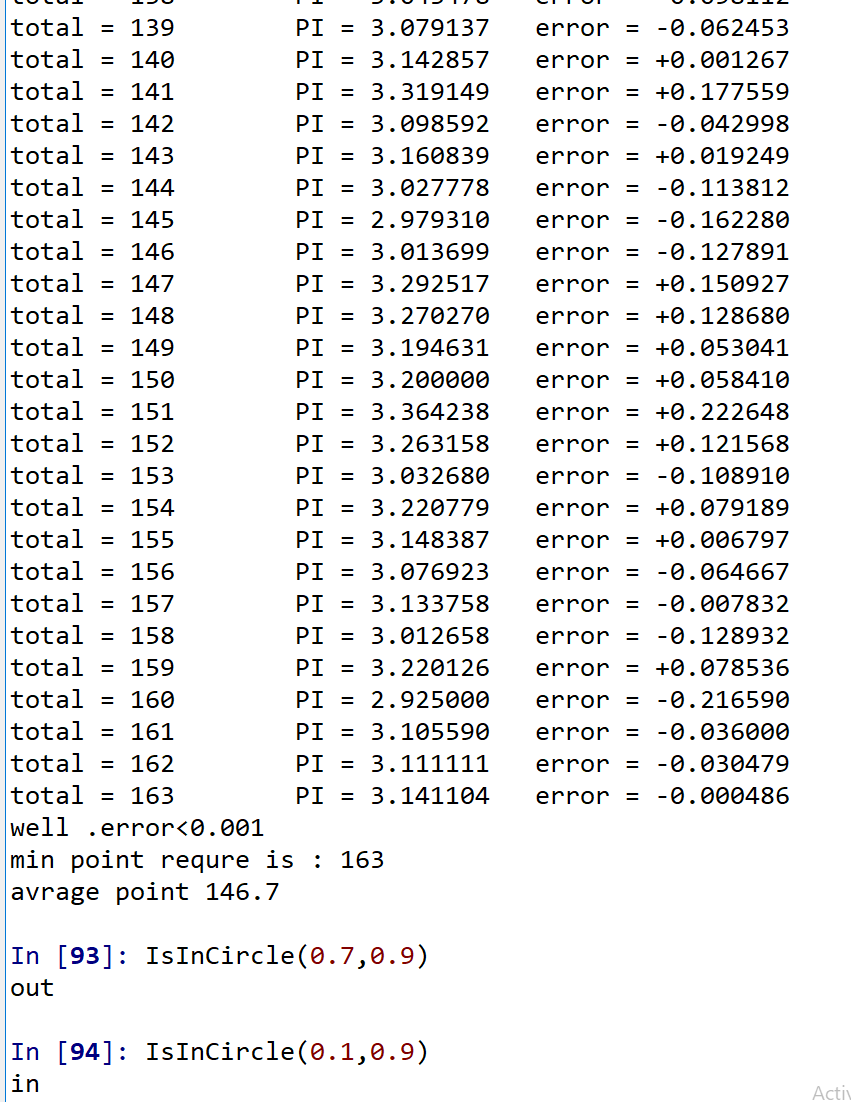
میخواهیم مقدار عدد پی را به دست آوریم که به این منظور یک دایره را درون یک مربع به ضلع یک محیط میکنیم .عدد پی برابر میشود با مساحت دایره به مساحت مربع که در واقع نقاطی را به صورت تصادفی برایشان تعریف میکنیم و اگه این اعداد داخل دایره قرار گیرند به مقدار نقاط داخل دایره یکی اضافه میکنیم و این روند را تا پایان کل تعداد نقاط ادامه میدهیم تا به عدد پی برسیم.

ابتدا طبق خواسته ی سوال تابع IsInCircle(x,y) را تعریف میکنیم ک به عنوان ورودی دو نقطه بین 0 و 1 را میگیرد و در خروجی نشان میدهد که آیا نقطه داخل دایره قرار دارد یا خیر.

سپس طبق خواسته سوال پیش رفته و تابع find() را جهت پیدا کردن مینیموم نقاط مورد نیاز در کل مساحت مربع مینویسیم به صورتی که این نقاط به ما خطای یک درصد را بدهند.پس یک error نیز تعریف میکنیم تا با مقدار واقعی عدد پی تا اعشار پنجم آن مقایسه شود و روند ادامه یابد تا خطای ما به کمتر از یک درصد برسد.

از آنجایی که با هر بار اجرا و گرفتن اعداد تصادفی، نقاط فرق میکنند و سرعت رسیدن به عدد پی با دقت دلخواه متفاوت میباشد، از ورودی عددی جهت تعداد تکرار روند و گرفتن میانگین از تعداد نقاط مورد نیاز، دریافت میکنیم و در نهایت در خروجی میانگین کمترین تعداد نقاط مورد نیاز را چاپ میکنیم.

نمونه خروجی با تابع find(10) و همچنین تابع IsInCorcle(x,y) آورده شده :

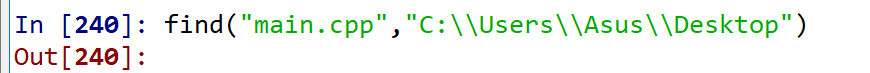


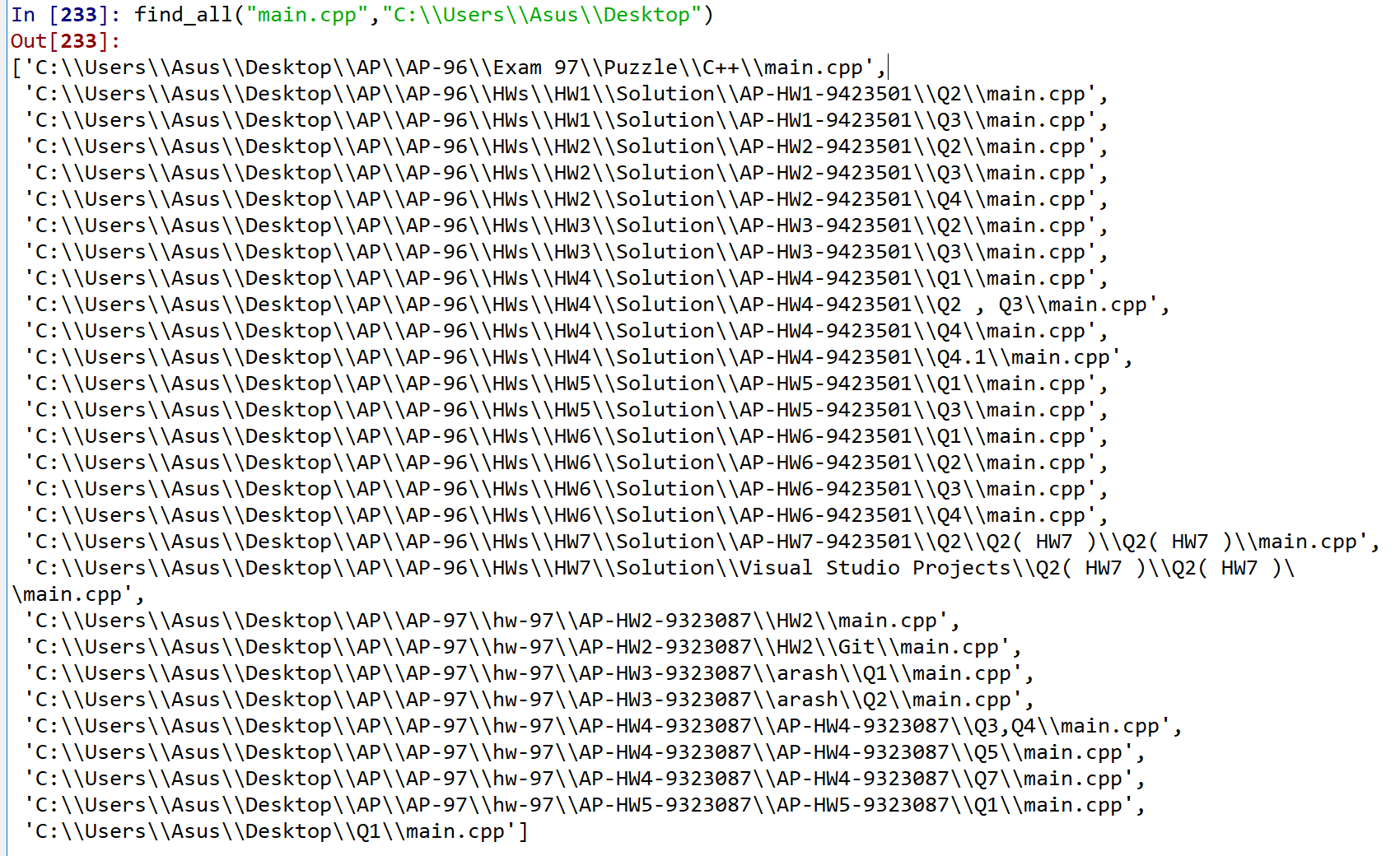
**سوال5:**

هر کدام از بخش های خواسته ی سوال به صورت مجزا در کد زیر آورده شده است که به وضوح بیانگر نحوه ی کار آن می باشد.برای نمونه خروجی یکی از توابع نیز ضمیمه گشته است:



همانطور که مشاهده میشود آدرس تمامی فایل هایی که اسم آنها main.cpp است را در یک لیست برمیگرداند:





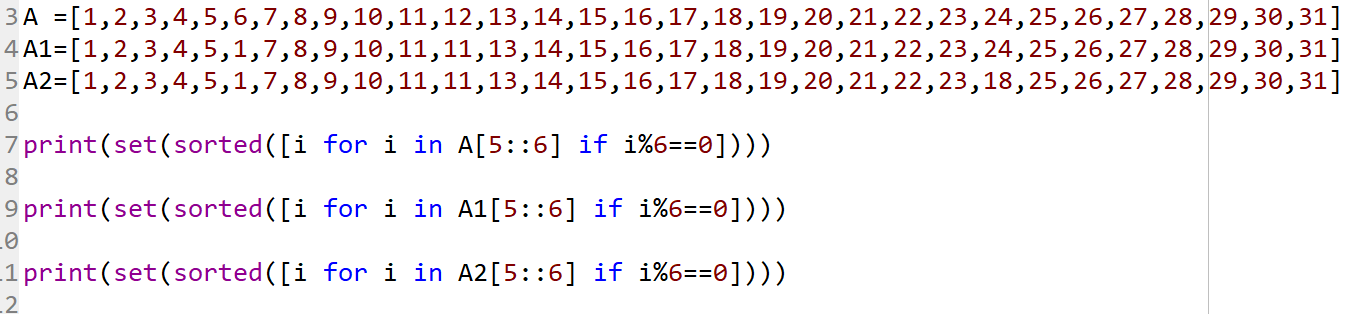
**سوال6:**

برای حل انتگرال این سوال ، ابتدا بازه های انتگرال و یک عدد به صورت عمومی تعریف شده، سپس کلاس گوس و توابع لژاندر، مشتق لژاندر، تابع پیداکردن صفر چند جمله‏ای لژاندر، تابع وزن،quadrature ، خود تابع اصلی و تابع time برای اندازه‏گیری زمان تعریف شده است. در توابع لژاندر ابتدا یک لیست خالی تعریف شده که این لیست به وسیله تابع بازگشتی مقادیر چند جمله‏ای لژندر را برمی گرداند. سپس مشتق تابع لژاندر نیز به وسیله یه لیست تعریف شده که با استفاده از تابع بازگشتی مقدارش برگردانده می‏شود. وزن تابع نیز به وسیله تابع بازگشتی تعیین شده است. در تابع ای که ریشه ها حساب می‏شوند؛ ابتدا بررسی می‏شود که بازه ی انتگرال بین -1 و 1 است یا نه. اگر نبود به وسیله ی فرمول مربوطه بازه را تغییر داده و سپس انتگرال به وسیله تابع اصلی و متغییر های تولید شده حساب می‏شود.

**سوال 7:**

در این سوال خواسته شده که ارایه ورودی را گرفته و درایه هایی را که مضارب عدد شش هستند چک کنیم و اگر این درایه ها خودشان حاوی عددی از مضارب شش بودند آنها را در خروجی نمایش دهیم.تمام این کار ها باید در یک خط انجام شود و عدد تکراری نیز در خروجی وجود نداشته باشد.بدین منظور از کد زیر استفاده کردیم:

print(set(sorted([i for i in A[5::6] if i%6==0])))

ابتدا تمامی درایه های ضرایب عدد 6 را با نوشتن A[5::6] مشخص میکنیم و با یک حلقه تمام آنهایی را که علاوه بر درایه مضرب 6 خودشان نیز باقیمانده ی صفر از تقسیم بر 6 دارند و بر شش بخش پذیر هستند را مشخص مینماییم و با دستورsorted آنها را از کوچک به بزرگ مرتب میسازیم .حال میتوانیم با دستور set خرجی را از هرگونه تکرار پاک کنیم. بدین منظور میتوانیم آرایه ی A را با مقادیر متفاوت بدهیم و خروجی مورد نظر را بررسی کنیم: 

مشاهده میشود ک پاسخ ها درست هستند؛ خروجی:

