

## گزارش تمرین سری سوم درس برنامه نویسی پیشرفته

### محمد صالح گشانی – 9423092

در کد های فرستاده شد برای فهم بهتر کد سعی شد تا جای ممکن برای دستورات کامنت گذاری شود .

#### سوال اول (C++)

1 – move semantics : در برنامه نویسی دو نوع مقدار وجود دارد Rvalue و Lvalue . برای مثال در تساوی  $x, x = y + 4$  یک Lvalue و  $y + 4$  یک Rvalue است . مقادیر Rvalue پس از پایان دستور از بین می روند و نمی توان از آن ها استفاده کرد و همچنین نمی توان آن ها را به صورت معمول در توابع استفاده کرد . برای استفاده از مقادیر Rvalue در کلاس ها و توابع راه و رسم و syntax خاصی وجود دارد که به این مجموعه move semantics گفته می شود .

2 – polymorphism : به معنی داشتن شکل های متفاوت است و معمولاً زمان هایی رخ می دهد که تعدادی کلاس که از همدیگر ارث می برند ، وجود داشته باشد . polymorphism به این معنی است که اگر یک تابع را از یک کلاس (کلاس والد) صدا کنیم ، با توجه به این که صدا کننده ی آن تابع شی از چه کلاسی (کلاس والد و یا کلاس فرزند) باشد نتایج متفاوتی خواهیم داشت . در واقع می توانیم تعدادی کلاس متفاوت داشته باشیم که همه ی آن ها تابعی با اسم یکسان و متغیر های یکسان داشته باشند ولی کار متفاوتی انجام دهند و با توجه به این که شی کدام کلاس تابع را صدا می زند خروجی متفاوت خواهد بود (لازم به ذکر است که برای اینکه هنگام صدا زدن تابع ، تابع مربوط به همان کلاس صدا زده شود باید تابع به صورت virtual تعریف شده باشد ) .

3 – pure abstract : کلاس abstract کلاسی است که در آن یک تابع که به صورت pure virtual (تابعی که پیشوند virtual را دارد و هنگام declare شدنش در فایل h. مساوی صفر قرار داده شده است) تعریف شده است وجود دارد . یکی از ویژگی های چنین کلاسی این است که نمی توان یک شی از آن ساخت . کلاسی که تنها توابع به صورت pure virtual دارد و توابع دیگری ندارد کلاس pure abstract است . این کلاس ها معمولاً متغیر ندارند .

4 – override : override کردن توابع به ما اجازه می دهد که در کلاس فرزند تابعی تابعی همانند یک تابع در کلاس والد تعریف کنیم در حالی که تابع جدید می تواند پیاده سازی متفاوتی با تابع کلاس والد داشته باشد و در بیرون از کلاس از تابع جدید استفاده کنیم .

5 – inline : عبارت inline در مفهوم کلاس به متغیر ها و توابعی گفته می شود که در درون تعریف کلاس (در ساختاری که در درس برنامه نویسی پیشرفته استفاده می کنیم تعریف کلاس در فایل با پسوند h. است ) تعریف و مقدار دهی می شوند . توابع inline علاوه بر این که در تعریف کلاس declare می شوند در همانجا نوشته هم می شوند و در فایل با پسوند cpp. جایی ندارند .

6 – explicit : در کلاس ممکن است ما یک constructor داشته باشیم که تنها یک ورودی برای مثال یک double (برای مثال این ورودی می تواند شعاع در کلاس دایره باشد) بگیرد . در ادامه ممکن است یک تابع داشته باشیم که یک شی از کلاس را

به عنوان ورودی بگیرد (برای مثال یک تابع که اندازه ی مساحت را در کلاس دایره مقایسه کند). حال اگر به ورودی تابع به جای یک دایره یک عدد بدهیم و که منظورمان از آن عدد مساحت باشد، از آنجا که کلاس یک constructor با یک ورودی دارد آن عدد را به عنوان شعاع دایره برداشته و یک دایره با مساحت جدید (که با مساحت مورد نظر ما فرق دارد) تولید می کند و خروجی به دست آمده از تابع اشتباه می شود. برای رفع این مشکل باید constructor را با پیشوند explicit تعریف کنیم. با این کار می توان از implicit conversion های از این قبیل جلوگیری کرد. همچنین در صورت استفاده از تابع ذکر شده با ورودی double یک خطا دریافت می کنیم.

## سوال دوم (C++)

در این سوال هنگام نوشتن تابع display که size و capacity وکتور را نشان می دهد باید وکتور را pass by reference می کردیم چرا که unique\_ptr تنها می تواند به یک شی اشاره کند و همچنین نمی تواند کپی شود. همچنین هنگام push\_back کردن مقادیر جدید باید از دستور std::make\_unique استفاده می کردیم که با این دستور کاری می کردیم که به متغیر جدید تنها یک پوینتر که همان unique\_ptr است اشاره داشته باشد. در مرحله ی اول که 1000 خانه را برای وکتور رزرو نکرده بودیم همانطور که از نتایج شکل 1 قابل مشاهد است با افزایش size، capacity هم به مقادیر random افزایش می یابد و هر بار که size با capacity برابر می شود، capacity مقدار جدیدی پیدا می کند در حالی که در مرحله ی دوم که 1000 خانه رزرو می شد می بینیم که capacity همواره 1000 مانده و size با هر push\_back اضافه می شود (شکل 2).

کد هر دو قسمت در فایل main موجود است و کافی است برای تست هر قسمت کد مربوط به آن قسمت uncomment شود.

```

garch@ubuntu: ~/Desktop/HW4/Q2
File Edit View Search Terminal Help
garch@ubuntu:~/Desktop/HW4/Q2$ make
g++ -std=c++17 -Wall -c main.cpp
g++ -std=c++17 main.o -o main
garch@ubuntu:~/Desktop/HW4/Q2$ ./main
Size :1 and Capacity : 1
Size :2 and Capacity : 2
Size :3 and Capacity : 4
Size :4 and Capacity : 4
Size :5 and Capacity : 8
Size :6 and Capacity : 8
Size :7 and Capacity : 8
Size :8 and Capacity : 8
Size :9 and Capacity : 16
Size :10 and Capacity : 16
Size :11 and Capacity : 16
Size :12 and Capacity : 16
Size :13 and Capacity : 16
Size :14 and Capacity : 16
Size :15 and Capacity : 16
Size :16 and Capacity : 16
Size :17 and Capacity : 32
Size :18 and Capacity : 32
Size :19 and Capacity : 32
Size :20 and Capacity : 32
Size :21 and Capacity : 32
Size :22 and Capacity : 32
Size :23 and Capacity : 32
Size :24 and Capacity : 32
Size :25 and Capacity : 32
Size :26 and Capacity : 32
Size :27 and Capacity : 32
Size :28 and Capacity : 32
Size :29 and Capacity : 32
Size :30 and Capacity : 32
Size :31 and Capacity : 32
Size :32 and Capacity : 32
Size :33 and Capacity : 64
Size :34 and Capacity : 64
Size :35 and Capacity : 64
Size :36 and Capacity : 64
Size :37 and Capacity : 64
Size :38 and Capacity : 64
Size :39 and Capacity : 64
Size :40 and Capacity : 64
Size :41 and Capacity : 64
Size :42 and Capacity : 64
Size :43 and Capacity : 64
Size :44 and Capacity : 64
Size :45 and Capacity : 64
Size :46 and Capacity : 64
Size :47 and Capacity : 64
Size :48 and Capacity : 64
Size :49 and Capacity : 64
Size :50 and Capacity : 64
Size :51 and Capacity : 64
Size :52 and Capacity : 64

```

شکل 1 – خروجی بخش اول سوال 2

```
garch@ubuntu: ~/Desktop/HW4/Q2$ make
g++ -std=c++17 -Wall -c main.cpp
g++ -std=c++17 main.o -o main
garch@ubuntu:~/Desktop/HW4/Q2$ ./main
Size :1 and Capacity : 1000
Size :2 and Capacity : 1000
Size :3 and Capacity : 1000
Size :4 and Capacity : 1000
Size :5 and Capacity : 1000
Size :6 and Capacity : 1000
Size :7 and Capacity : 1000
Size :8 and Capacity : 1000
Size :9 and Capacity : 1000
Size :10 and Capacity : 1000
Size :11 and Capacity : 1000
Size :12 and Capacity : 1000
Size :13 and Capacity : 1000
Size :14 and Capacity : 1000
Size :15 and Capacity : 1000
Size :16 and Capacity : 1000
Size :17 and Capacity : 1000
Size :18 and Capacity : 1000
Size :19 and Capacity : 1000
Size :20 and Capacity : 1000
Size :21 and Capacity : 1000
Size :22 and Capacity : 1000
Size :23 and Capacity : 1000
Size :24 and Capacity : 1000
Size :25 and Capacity : 1000
Size :26 and Capacity : 1000
Size :27 and Capacity : 1000
Size :28 and Capacity : 1000
Size :29 and Capacity : 1000
Size :30 and Capacity : 1000
Size :31 and Capacity : 1000
Size :32 and Capacity : 1000
Size :33 and Capacity : 1000
Size :34 and Capacity : 1000
Size :35 and Capacity : 1000
Size :36 and Capacity : 1000
Size :37 and Capacity : 1000
Size :38 and Capacity : 1000
Size :39 and Capacity : 1000
Size :40 and Capacity : 1000
Size :41 and Capacity : 1000
Size :42 and Capacity : 1000
Size :43 and Capacity : 1000
Size :44 and Capacity : 1000
Size :45 and Capacity : 1000
Size :46 and Capacity : 1000
Size :47 and Capacity : 1000
Size :48 and Capacity : 1000
Size :49 and Capacity : 1000
Size :50 and Capacity : 1000
Size :51 and Capacity : 1000
Size :52 and Capacity : 1000
```

شکل 2 – خروجی بخش دوم سوال 2

## سوال سوم (C++)

در حل سوال 3 ابتدا یک کلاس به نام Point تعریف کردیم که این کلاس 3 متغیر داشت که نشان دهنده ی مختصات آن نقطه بودند . سپس کلاس Shape را تعریف کردیم که از آنجا که این کلاس قرار بود یک کلاس pure abstract باشد سه تابع `print` , `volume` , `area` را به صورت pure virtual برای آن تعریف کردیم همچنین اپراتور << را نیز به صورت friend و درون تعریف خود کلاس برای آن تعریف کردیم . friend تعریف کردن باعث می شود که با اینکه تابع خارج از کلاس است ولی به محتویات کلاس دسترسی داشته باشد و ما نیاز داشتیم که به تابع `print` شی کلاس Shape دسترسی داشته باشیم .

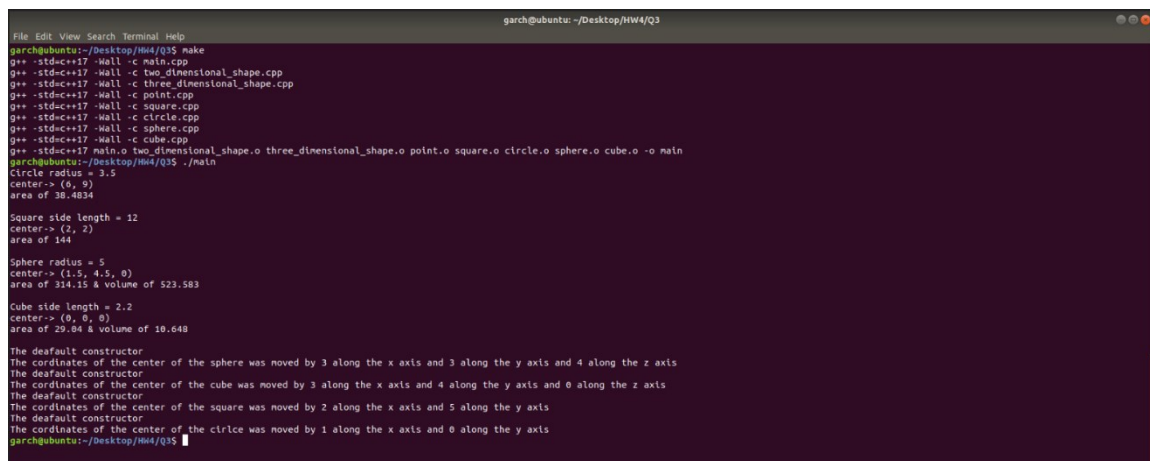
در مرحله ی بعد کلاس های `TwoDimensionalShape` و `ThreeDimensionalShape` را به صورت ارث برده از کلاس Shape ساختیم . در کلاس 2D از آنجا که اشکال دو بعدی هستند دو متغیر که برای مختصات دو بعدی بودند اختصاص دادیم . در کلاس 3D هم سه متغیر اختصاص دادیم . همچنین این دو کلاس همانند کلاس Shape به صورت abstract بودن و توابع `print` , `volume` , `area` همانند کلاس Shape به صورت pure virtual در این دو کلاس تعریف شد . علاوه بر این برای هر دو کلاس constructor های متعدد با توجه به تعداد مختصات مورد نیاز تعریف شد .

در مرحله ی کلاس اشکال مختلف دایره ، مربع ، مکعب و کره را با توجه به دو بعدی و یا سه بعدی بودن ، به صورت ارث برده از یکی از دو کلاس 2D یا 3D ساختیم . سپس در constructor های این کلاس ها با توجه به کلاس والدشان ابتدا مختصات دو یا سه بعدی کلاس والد را مقداردهی کرده سپس متغیر مخصوص کلاس (برای دایره و کره شعاع و برای مربع و کره ضلع) را مقدار دهی کردیم . سپس توابع `volume` , `area` را با توجه به دو و یا سه بعدی بودن پیاده سازی کردیم . به این صورت که اگر در کلاس های دو بعدی مقدار حجم 0 برگردانده شده و برای محاسبه ی سایر مساحت ها و حجم ها با توجه به فرمول های محاسبه ی هر کدام متناسب با شکل ، این مقادیر محاسبه و بازگردانده شدند . سپس تابع `print` را هم برای هر کلاس تعریف کردیم که

در هر کلاس مختصات مرکز و همچنین اندازه ی متغیر مخصوص و مساحت و حجم (در کلاس های دایره و مربع فقط مساحت) را چاپ می کرد .

همچنین برای این کلاس ها اپراتور << را تعریف کردیم . که در این اپراتور تابع print اجرا می شد . در آخر هم اپراتور + را در هر کلاس تعریف کردیم . از آنجا که نیاز بود خروجی اپراتور + همان کلاس باشد تا در صورت نیاز بتوانیم مختصات مرکز یک شکل مشابه را تغییر دهیم نمی توانستیم این اپراتور را روی پوینتر ها اجرا کنیم چرا که در این صورت نیاز بود که اپراتور + برای کلاس های Shape و همچنین 2D و 3D نیز تعریف کنیم و از آنجا که این 3 کلاس abstract بودند نمی شد خروجی یک اپراتور را از نوع abstract تعریف کنیم . به همین دلیل اپراتور ها را به طور جدا برای هر یک از 4 کلاس دایره و مربع و مکعب و کره تعریف کرده و برای تست هم اپراتور + را روی متغیر های ساخته شده در main یعنی cir , cub , sph , sqr امتحان کرده و نتایج را چاپ کردیم .

یک نمونه از خروجی در شکل 3 قابل مشاهده است .



```
garch@ubuntu:~/Desktop/HW4/Q3$ make
g++ -std=c++17 -Wall -c main.cpp
g++ -std=c++17 -Wall -c two_dimensional_shape.cpp
g++ -std=c++17 -Wall -c three_dimensional_shape.cpp
g++ -std=c++17 -Wall -c point.cpp
g++ -std=c++17 -Wall -c square.cpp
g++ -std=c++17 -Wall -c circle.cpp
g++ -std=c++17 -Wall -c sphere.cpp
g++ -std=c++17 -Wall -c cube.cpp
g++ -std=c++17 main.o two_dimensional_shape.o three_dimensional_shape.o point.o square.o circle.o sphere.o cube.o -o main
garch@ubuntu:~/Desktop/HW4/Q3$ ./main
Circle radius = 3.5
center-> (5, 9)
area of 38.4834

Square side length = 12
center-> (2, 2)
area of 144

Sphere radius = 5
center-> (1.5, 4.5, 0)
area of 314.15 & volume of 523.583

Cube side length = 2.2
center-> (0, 0, 0)
area of 29.84 & volume of 10.648

The default constructor
The coordinates of the center of the sphere was moved by 3 along the x axis and 3 along the y axis and 4 along the z axis
The default constructor
The coordinates of the center of the cube was moved by 3 along the x axis and 4 along the y axis and 0 along the z axis
The default constructor
The coordinates of the center of the square was moved by 2 along the x axis and 5 along the y axis
The default constructor
The coordinates of the center of the circle was moved by 1 along the x axis and 0 along the y axis
garch@ubuntu:~/Desktop/HW4/Q3$
```

شکل 3 – خروجی سوال 3

## سوال چهارم (C++)

برای تعریف تابع print باید از پسوند virtual استفاده شود چرا در هر تابع print عبارات خاص مربوط به هر شکل خاص نوشته شده و اگر می خواهیم برای هر شکل تعاریف درست چاپ شود باید برای هر شکل توابع متناسب با آنها اجرا شود علاوه بر این در برخی از print ها ما volume را محاسبه می کنیم و در برخی دیگر این کار را نمی کنیم که این هم خود باعث تمایز بین print هاست . همچنین دو تابع area و volume نیز باید به صورت virtual تعریف شوند چرا که حجم و سطح مربوط به هر شکل نحوه ی محاسبه (پایاده سازی) خاصی دارد و هنگام اجرا شدن آنها باید با توجه به نوع شکل این پارامتر ها محاسبه شوند .

در حالت کلی زمانی از پیشوند virtual استفاده می کنیم که کلاس والد و کلاس فرزند وجود داشته باشد و در کلاس (های) فرزند تابعی همانند یکی از توابع کلاس والد وجود داشته باشد که پیاده سازی این تابع با پیاده سازی تابع کلاس والد فرق کند و ما نیاز داشته باشیم که هنگام صدا زدن تابع کلاس فرزند توسط شی کلاس فرزند تابع کلاس والد اجرا شود . با استفاده از پیشوند virtual پشت تابع در تمامی کلاس ها می توان خروجی مطلوب را به دست آورد .

## سوال پنجم (C++)

برای حل این سوال از مثال کتاب برای تعریف کردن Stack استفاده کردیم . کتاب برای این کار از یک nested class استفاده کرد بود . به این صورت که در کلاس Stack یک کلاس Node تعریف شده که هر Node نشانگر یک ورودی Stack بود . هر Node یک مقدار داشته و یک پوینتر که به Node بعدی در Stack اشاره کند .

در کلاس Stack هم با هر بار اضافه کردن یک Node مقدار Head آن عوض شده و برابر با Node جدید قرار داده می شد و Node قبلی که برابر با Head بود در پوینتر Head جدید ذخیره می شد . برای حل سوال هم چون باید یک تابع به نام getCount برای کلاس Stack تعریف می کردیم برای این کلاس یک متغیر به نام size تعریف کرده که با هر بار push شدن یکبار اضافه شده و با هر بار pop یکبار کم شود و در تابع getCount این مقدار باز گردانده شود .

همچنین برای حل سوال یک کلاس به نام CText تعریف کرده که این کلاس یک متغیر به نام text از نوع string داشته که با اجرای تابع getText روی یک شی از این کلاس text آن بازگردانده می شد .

یک نمونه از خروجی در شکل 4 قابل مشاهده است .

```
garch@ubuntu:~/Desktop/HW4/q55$ make
g++ -std=c++17 -Wall -c main.cpp
g++ -std=c++17 main.o -o main
garch@ubuntu:~/Desktop/HW4/q55$ ./main
TEXTZ
TEXTY
TEXTX
TEXTW
TEXTV
TEXTU
TEXTT
TEXTS
TEXTR
TEXTQ
TEXTP
TEXTO
TEXTN
TEXTM
TEXTL
TEXTK
TEXTJ
TEXTI
TEXTH
TEXTG
TEXTF
TEXTE
TEXTD
TEXTC
TEXTB
TEXTA
stack is empty
garch@ubuntu:~/Desktop/HW4/q55$
```

شکل 4 – خروجی سوال 5

## سوال ششم (C++)

برای این سوال ابتدا یک تابع به نام display با template نوشتیم که در آن با استفاده از range based for loop مقادیر container ها را در هر مرحله چاپ کنیم . همچنین در ابتدا هر بخش وکتور اولیه را بازیابی کردیم تا کار های انجام شده رو همان بردار اولیه انجام شود .

برای بخش الف ابتدا دستور std::remove را روی وکتور اجرا کردیم . با این دستور مقداری که می خواهیم حذف کنیم به انتهای وکتور منتقل شده و iterator آن به عنوان خروجی داده می شود . سپس با دستور erase مقادیر بعد از iterator به دست آمده که همان 2 های وکتور بودند را از وکتور حذف کردیم .

برای بخش ب از یک functor استفاده کردیم که مقدار هر المان آرایه را دو برابر کند .

برای بخش ج ابتدا با تابع `std::accumulate` مقدار میانگین را حساب کردیم . سپس با فرستادن یک تابع لاند که دو ورودی گرفته و اگر فاصله ی ورودی اول از میانگین بیشتر از فاصله ی ورودی دوم از میانگین بود ، `True` بر می گرداند ، به تابع `std::sort` ، وکتور را برحسب فاصله از میانگین مرتب کردیم .

برای بخش د ابتدا با دستور `sort` مقادیر وکتور را از کوچک به بزرگ مرتب کردیم . سپس با استفاده از دستور `std::unique` کاری کردیم که ابتدا از هر یک از المان ها فقط یک عدد وجود داشته باشد و المان هایی که تکرار می شدند ، مقادیر تکراری شان به انتهای وکتور منتقل شود . در نهایت هم با استفاده از دستور `erase` مقادیر تکراری را حذف کردیم .

برای بخش ر هم پس از ریختن وکتور درون یک `set` ، با استفاده از خاصیت ترتیبی بودن یک `set` (از کوچک به بزرگ مرتب می شود) و استفاده از دستور `std::find` ابتدا `iterator` عدد 3 را پیدا کرده سپس اعداد 4 به بعد را از `set` حذف کردیم .  
یک نمونه از خروجی در شکل قابل مشاهده است .

```
File Edit View Search Terminal Help
garch@ubuntu:~/Desktop/HW4/Q6$ make
g++ -std=c++17 -Wall -c main.cpp
g++ -std=c++17 main.o -o main
garch@ubuntu:~/Desktop/HW4/Q6$ ./main
1 2 3 4 5 4 3 2 1
The vector vec after rearranging contains :
1 3 4 5 4 3 1 2 1
The vector vec after removing 2s contains :
1 3 4 5 4 3 1
The vector vec after being doubled contains :
2 4 6 8 10 8 6 4 2
The average of the vec is : 2.77778
The vector vec after sorting based on the distance from the average, contains :
5 1 1 4 4 2 2 3 3
The vector vec after erasing repeated elements contains :
1 2 3 4 5
The set SET after being initialized with vec contains :
0 1 2 3 4 5
The set SET after removing elements bigger than 3 contains :
0 1 2 3
garch@ubuntu:~/Desktop/HW4/Q6$
```

شکل 6 – شکل خروجی سوال 6

## سوال هفتم (C++)

برای این سوال ابتدا سه وکتور به طول 50 به نام های `a` و `b` و `c` ساختیم . سپس با دستور `std::generate` و با کمک یک تابع لاند `b` را با مقادیر تصادفی مقدار دهی کردیم . سپس با دستور `sort` وکتور `b` را به صورت صعودی مرتب کرده سپس دستور `std::unique` را روی آن اجرا کردیم . با اجرای دستور `std::unique` ابتدا تمام ارقام موجود در وکتور به صورت صعودی و با فراوانی یک چیده شده و پس از چیده شدن آخرین مقدار ، اعدادی در وکتور اولیه ی `b` بیش از 1 بار تکرار شده بودند تکرار دوم به بعدشان چیده می شود و `iterator` شروع این تکرار ها (که همان پایان اعداد چیده شده با فراوانی 1 است) به عنوان خروجی بیرون داده می شود . در نهایت هم با دستور `erase` اعداد شروع شده از `iterator` خروجی دستور `std::unique` تا پایان وکتور را پاک کردیم .

سپس از آنجا که وکتور `c` نیز دارای 50 المان بوده و نیاز بود در ادامه با وکتور `b` اعمالی روی هم انجام دهند لازم شد که با استفاده از دستور `resize` اندازه ی وکتور `b` را 50 بکنیم (با اضافه کردن 0 در انتهای آن) . سپس با دستور `std::iota` و با شروع از 1 مقدار 1 تا 50 را به صورت صعودی در المان های وکتور `c` قرار دادیم . در آخر هم با استفاده از دستور `std::transform` و

برای نمایش المان های وکتور ها بدون استفاده از حلقه از دستور `std::copy` استفاده کردیم . به این صورت که با دادن یک `std::ostream_iterator` به عنوان ورودی سوم `std::copy` مقادیر وکتور را چاپ کردیم .

[illegible]

لازم به ذکر است که برای اضافه کردن سوال ها و گزینه ها از رابط کاربری **admin** استفاده کرده و با استفاده از کد این کار را انجام ندادیم .

در شکل های زیر می توانید نمونه هایی از برنامه ی نهایی را مشاهده کنید .

Add question

Question text:

Date information (Show)

CHOICES		
CHOICE TEXT	VOTES	DELETE?
<input type="text"/>	0 <input type="button" value="↑"/>	<input type="button" value="X"/>
<input type="text"/>	0 <input type="button" value="↑"/>	<input type="button" value="X"/>
<input type="text"/>	0 <input type="button" value="↑"/>	<input type="button" value="X"/>

[+ Add another Choice](#)

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

رابط کاربری admin برای اضافه کردن سوال و جواب

✓ The question "Mood?" was added successfully.

Select question to change

Q  [Search](#)

Action:   0 of 2 selected

<input type="checkbox"/>	QUESTION TEXT	DATE PUBLISHED	PUBLISHED RECENTLY?
<input type="checkbox"/>	Mood?	May 7, 2019, 12:28 p.m.	✓
<input type="checkbox"/>	Weather?	May 9, 2019, 6 a.m.	✗

2 questions

[ADD QUESTION +](#)

**FILTER**

By date published

- Any date
- Today
- Past 7 days
- This month
- This year

سوال های اضافه شده همراه با زمان اضافه شدن

Select choice to change

[ADD CHOICE +](#)

Action:   0 of 6 selected

<input type="checkbox"/>	CHOICE
<input type="checkbox"/>	Neither
<input type="checkbox"/>	Up
<input type="checkbox"/>	Down
<input type="checkbox"/>	Windy
<input type="checkbox"/>	Rainy
<input type="checkbox"/>	Sunny

6 choices

جواب های موجود در جدول Choice

- [Mood?](#)

سوال موجود در صفحه ی ابتدایی polls



علت این که سوال Weather? در صفحه ی ابتدایی نمایش داده نشد اینست که زمان اضافه شدن آن سه روز بعد قرار داده شده است و تا زمانی که به زمان اضافه شدن آن نرسیم نمایش داده نمی شود .

## Mood?

☐ Down  
☐ Up  
☐ Neither

انتخاب های ممکن برای سوال Mood?

## Mood?

- Down -- 0 votes
- Up -- 1 vote
- Neither -- 2 votes

[Vote again?](#)

نتیجه ی نمایش داده شده پس از چند بار جواب دادن به سوال Mood?

## بارگذاری در Git

برای بارگذاری کد ها در github ، هنگام ساخت repository در قسمت آپشن ها فایل .gitignore را با تنظیمات ++C تشکیل دادیم که این باعث شد فایل های با پسوند نامطلوب خود به خود به این فایل اضافه شده و ما در ادامه فقط نام فایل main

را به آن اضافه کردیم . سپس در لینوکس دستور git init را اجرا کرده و سپس با دستور git clone repository را بعد از آن <https://github.com/MSalehG/AP-HW4> و بعد از آن git remote -v در یک دایرکتوری دلخواه repository را clone کردیم . در ادامه فایل های پوشه های برنامه های نوشته شده را در بخش clone شده اضافه کرده و با استفاده از دستور git add "folder" آن پوشه را در repository اضافه کردیم . سپس با دستور git commit -m "comment" فایل ها را برای بارگذاری آماده کردیم . در نهایت هم با دستور git push origin master فایل ها را فرستادیم . لازم به ذکر است که اگر نیاز به تغییرات در هر یک از فایل های commit شده بود تنها نیاز بود که فایل را ویرایش کرده و سپس دوباره آنرا add سپس commit و در نهایت push کنیم و فایل مورد نظر در repository خود به خود به روز رسانی می شد .

در شکل زیر می توانید مراحل انجام کار را مشاهده کنید .

```
garch@ubuntu:~/Desktop/Git/AP-HW4$ git add Q5
garch@ubuntu:~/Desktop/Git/AP-HW4$ git commit -m "second commit"
[master 0b702ca] second commit
1 file changed, 1 insertion(+), 1 deletion(-)
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
garch@ubuntu:~/Desktop/Git/AP-HW4$ git push origin master
Username for 'https://github.com': MSalehG
Password for 'https://MSalehG@github.com':
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/MSalehG/AP-HW4/'
garch@ubuntu:~/Desktop/Git/AP-HW4$ git push origin master
Username for 'https://github.com': MSalehG
Password for 'https://MSalehG@github.com':
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 3.85 KiB | 3.85 MiB/s, done.
Total 13 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MSalehG/AP-HW4
ea5930b..0b702ca master -> master
garch@ubuntu:~/Desktop/Git/AP-HW4$
```

دستورات اجرا شده برای بارگذاری در git

همچنین با کلیک روی لینک زیر می توانید به repository ساخته شده در git بروید .

<https://github.com/MSalehG/AP-HW4>