****

**دانشگاه صنعتی امیرکبیر**

**دانشکده برق**

**تمرین سری دوم**

**مریم ذاکری**

**9423047**

**سوال 1:**

**1.1:**

در این سوال با توجه به اینکه متغیرb پوینتری است که به آدرس متغیر a اشاره میکند وقتی محتوی این پوینتر را تغییر میدهیم در واقع مقدار a را عوض کرده ایم

در این کد یک بار در خط 5 و بار دیگر در خط 6 مقدار a یک واحد اضافه شده است ،با توجه به این که مقدار اولیه اش 10 بوده است در نهایت مقدارش به 12 میرسد .

مقدار b در خروجی برابر 0x7ffdae9a4f9c است که همان آدرس متغیر a است.

مقدار \*b نیز برابر a و برابر 12 است .

**1.2:**

**کد سوال**

**آیا کد کامپایل میشود ؟**

خیر .

Const int\* مقدار متغیر را const میکند و int\* const پوینتر را const میکند .

به این معنی که در حالت اول مقدار متغیری که پوینتر به آن اشاره میکند قابل تغییر نیست و در حالت دوم پوینتر نمی تواند به خانه ی دیگری اشاره کند .

در این سوال با توجه به اینکه در خط 10 مقدار پوینتر e که const است را برابر آدرس d قرار دادیم و در واقع پوینتر را تغییر داده کد با error مواجه میشود .

**1.3:**

**آیا کد کامپایل میشود ؟**

خیر

متغیر name در واقع پوینتری از جنس char\* است که به خانه ی اول عبارت درون آکولاد اشاره میکند .

با استفاده از این نوع تعریف کامپیلر name را به شکل آرایه ای از char میبیند که با استفاده از آرگومان اول آرایه به بقیه ی المانها نیز دسترسی پیدا میکند.

متغیر p1 که با name مقداردهی شده است نیز به همان آرایه ی کاراکتری اشاره میکند .

هر دو متعیر از جنس const char\* هستند به این معنی که محتویاتشان قابل تغییر نیست .

عبارت \*(p1) و \*(p1+1) به ترتیب به مقدار خانه ی اول آرایه و خانه ی دوم را نشان میدهند .

استفاده از عبارت &a در خط5 بلامانع است و \*(p1) مقدار این متغیر را به خروجی میدهد .

علت کامپایل نشدن کد این است که متغیر p1 را نمیتوان با ‘b’مجددا مقدار دهی کرد زیرا const char\* است و ما نمیتوانیم محتویاتش را مستقیم تغییر دهیم.

همچنین در خط 9 مجاز به استفاده از این کد نیستیم زیرا p2 const نیست در حالیکه،const, name است و همواره عبارت سمت چپ باید محدودتر از سمت راست تساوی باشد.

**1.4:**

**کد سوال**

**آیا کد کامپایل میشود؟**

**بله**

**متغیر p1 آرایه ی دینامیک است به طول 10 است .**

**متغیر p2 آرایه ای از پوینترها است .**

**P3پوینتر به تابع است .**

**سوال2**

**کد سوال**

در این سوال میخواهیم فایل txt ای را از طریق main به صورت خط به خط بخوانیم ،کلمات هر خط را جدا کنیم و با استفاده از ساختار شرطی if بررسی کنیم که آیا حروفی که کنار هم در یک کلمه قرار میگیرند چه وضعیتی دارند.

هدف سوال پیدا کردن غلطهای نگارشی ای است که در هر کلمه ی فایل اتفاق افتاده است .

برای این منظور باید چک کنیم که 5حرف متوالی غیر حروف صدادار ،حروف کوچک نباشند.

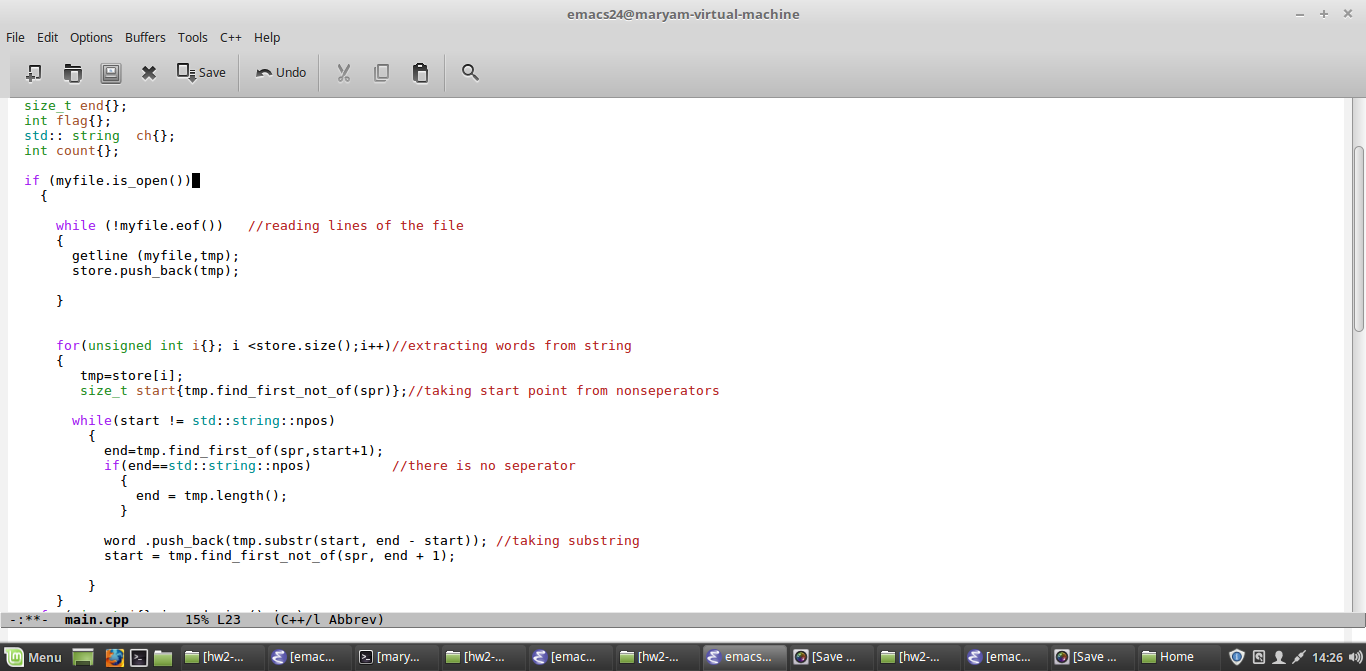
**الگوریتم به کاررفته در کد :**

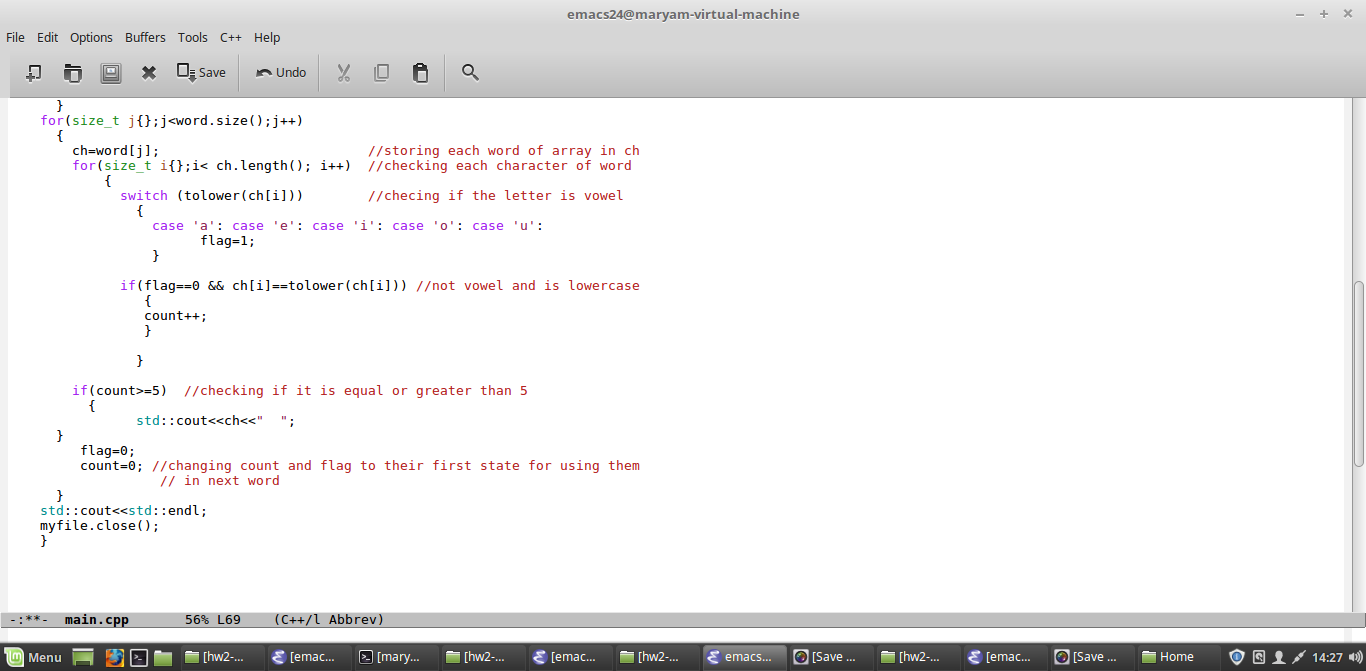
برای جدا کردن کلمات از دستور find\_first\_of و find\_first\_not\_of استفاده میکنیم که کلمات را با استفاده از جداکننده های {“ ,().”} و دستور بالا از یکدیگر تفکیک میکند.

این کد کراکترهایی را که بین دوجداکننده متوالی است با هم در نظر میگیرد .

با استفاده از یک vector از جنس stringاین متغیرها را به صورت کلمه ذخیره میکنیم .

در نهایت با استفاده از یک for تک تک کاراکترهای هر کلمه را از نظر صدار دار بودن و یا بزرگ و کوچک بودن بررسی میکنیم .با استفاده از یک متغیر count تعداد حروف غیر صدادار کوچک را که پشت سر هم آمده اند را ذخیره میکنیم ، اگر cout>=5 باشد این کلمه غلط نگارشی دارد در نتیجه آن را در خروجی نمایش میدهیم .



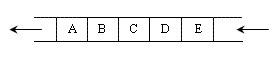


**سوال 3:**

**درباره ی منطق صف حلقوی**

صف لیست مرتبی است كه عناصر در انتهای آن (Rear) اضافه و از ابتدای آن(Front) حذف می شوند. به عبارت ديگر طول صف از انتهای آن افزایش و از ابتدای آن كاهش می یابد.

اولين عنصری که وارد صف می شود اولين عنصری است که از صف خارج می شود. بنابراين عناصر به همان ترتيبی که به صف اضافه می شوند از آن حذف می شوند. به همين دليل به صف لیست (first in, first out) FIFO نیز گفته می‌شود.



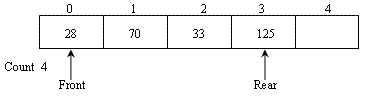
صف را می توان توسط يک آرايه يک بعدی پیاده سازی کرد. به دو متغیر Front و Rear برای مشخص كردن ابتدا و انتهای صف نياز است.

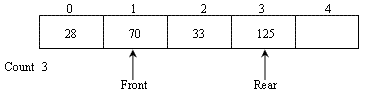
هر گاه عنصری به صف اضافه شود Rear یك گام به جلو حركت می كند و هر گاه كه عنصری را از صف حذف می شود Front یك واحد افزايش می يابد.

چون اندازه آرايه از قبل تعريف می شود، هنگام اضافه کردن عنصری به صف ابتدا باید اطمينان حاصل کرد که هنوز ظرفیت پذیرش داده را دارد. اگر Rear برابر با ظرفیت كل آرایه شود صف پر درنظر گرفته می شود.

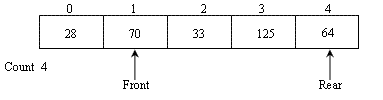
اگر ابتدا و انتهای صف برابر بودند (Front=Rear) یعنی صف خالی است. عمل حذف روی صف خالی انجام نمی گيرد.

طول صف يا تعداد عناصر موجود در صف برابر با Rear-Front+1 است.





**پاک کردن یک المان با جابجایی front صورت میپذیرد.**



**برای اضافه کردن المان به آرایه در نقطه ی بعدی ای که پوینتر rear به آن اشاره میکند مینویسیم .**

**برای حلقوی بودن باید در صورتی که پوینتر rear به خانه ی آخر رسید مجددا به خانه ی اول اشاره کند و اگر front هم به همان خانه اشاره میکرد اجازه ی نوشتن نداریم.**

**درباره ی الگوریتم کد:**

در این سوال میخواهیم کلاس Queue زا به گونه ای بنویسیم که فایلی را بخواند و با منطق صف حلقوی آرایه از اعداد درون فایل ایجاد کند .

با استفاده از method های این کلاس باید بتوانیم اعداد جدیدی را از کاربر دریافت کنیم و درون آرایه ی مورد نظر بنویسیم.

برای این منظور پوینترهای read و write را تعریف میکنیم که به شکل پیش فرض پس از خواندن فایل به ترتیب به آدرس خانه ی اول آرایه و اولین خانه ی خالی اشاره میکنند .

**enQueue(num) :** برای نوشتن در آرایه تابعی را تعریف میکنیم که محتوی خانه ای را که write به اشاره میکند را برابر مقداری قرار دهد که کاربر وارد میکند .

و هر بار پوینتر write را یک واحد جلو ببرد

در صورتی که به خانه ی اخر رسید مجددا به خانه ی اول اشاره کند .

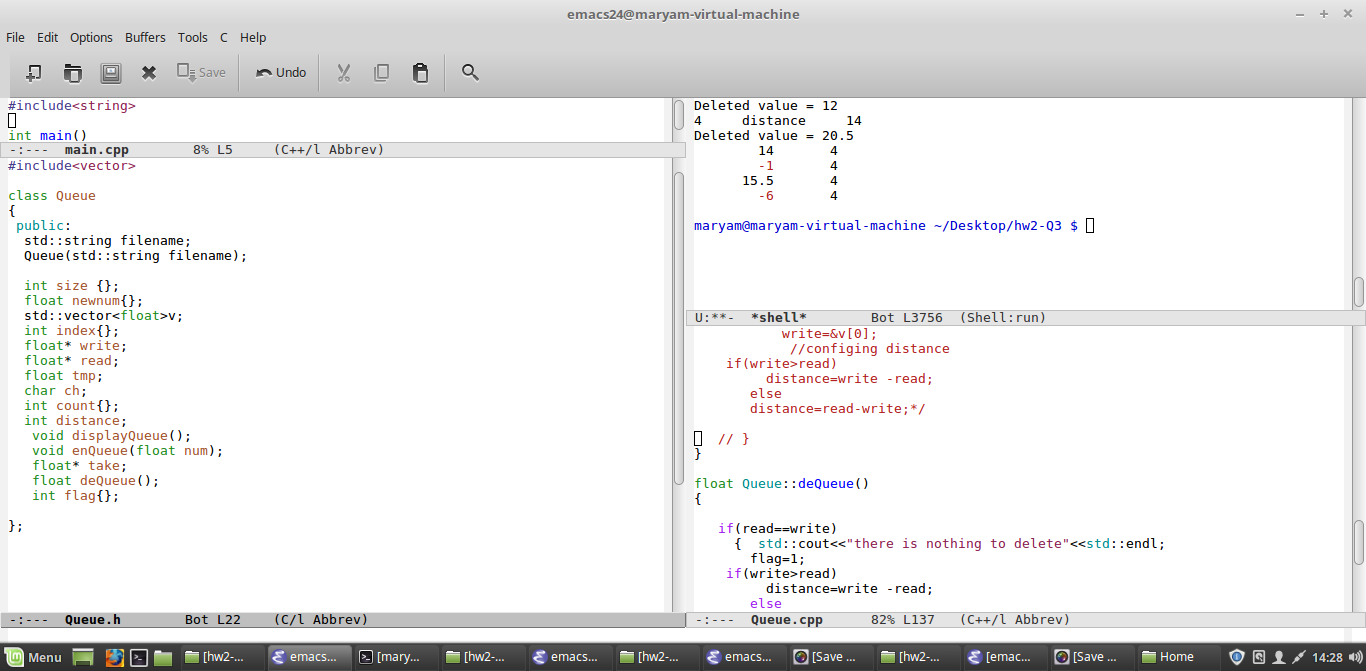
**deQueue() :** برای پاک کردن هر خانه باید پونتر read را جابجا کنیم و به خانه ی بعدی منتقل کنیم .

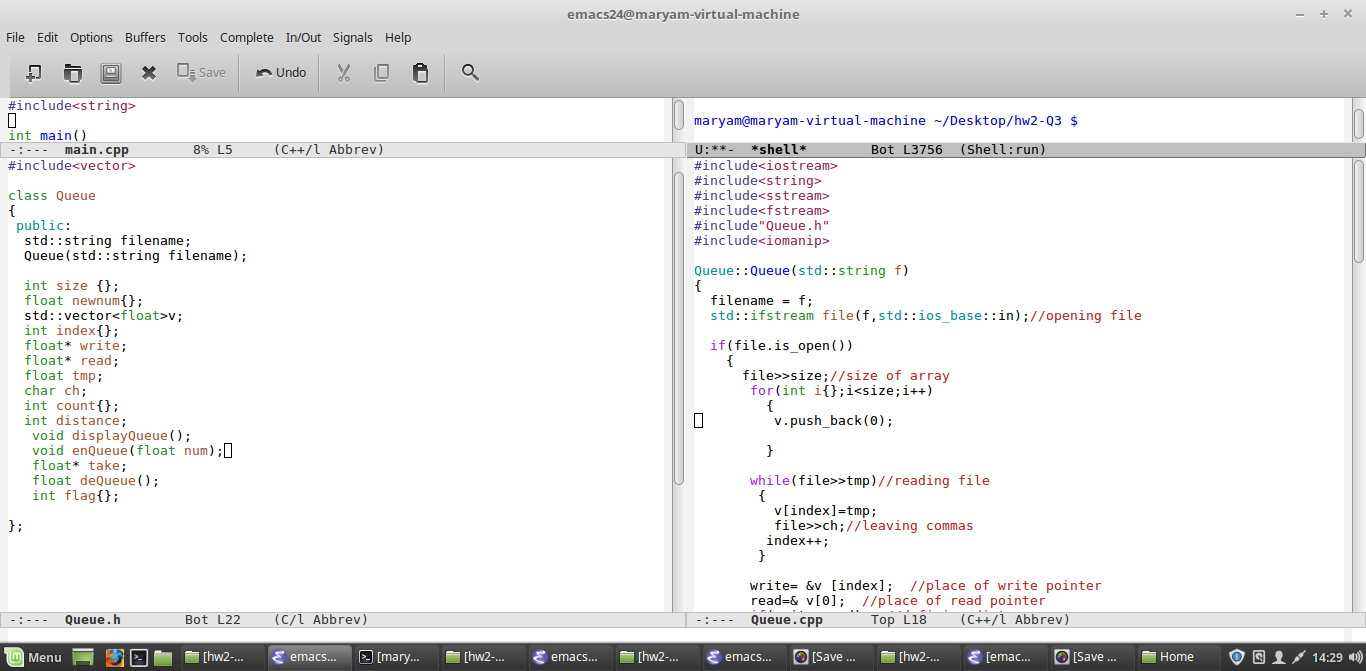
برای اینکار در تابع هربار مقدار متغیری را که read به آن اشاره میکند را return میکنیم و read++میکنیم .

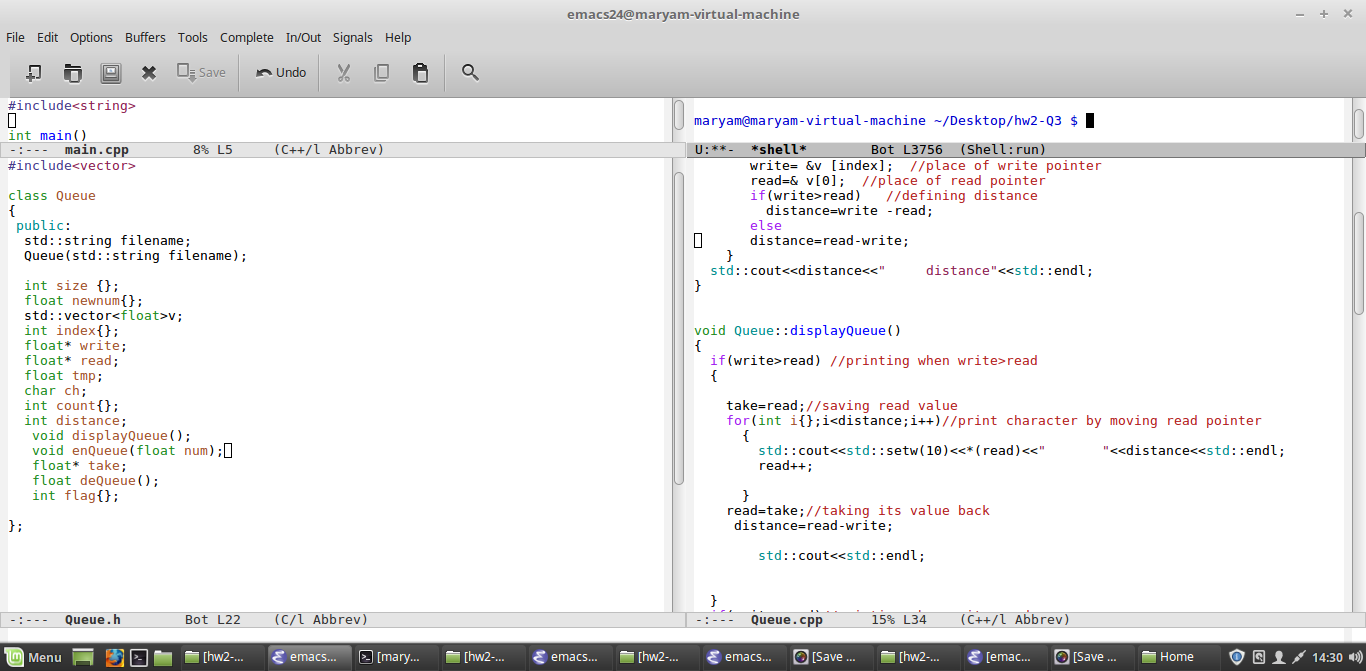
**displayQueue():** در این تابع هر بار آدرس write و read را از هم کم میکنیم و در متغیر distance ذخیره میکنیم

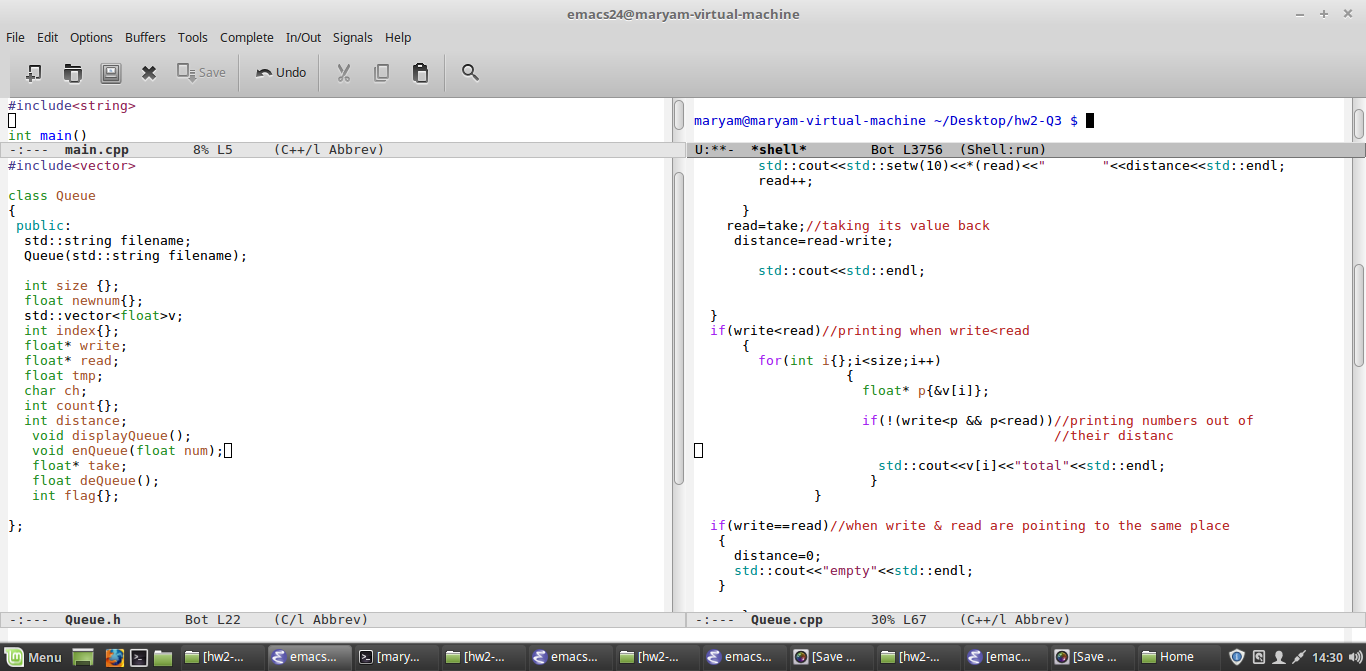
در صورتیکه write>read باشد محتوی read را نمایش میدهیم و یک واحد به آن اضافه میکنیم .و با استفاده از حلقه ی for به اندازه ی i=0 تا i<distance بار read را جلو میبریم.

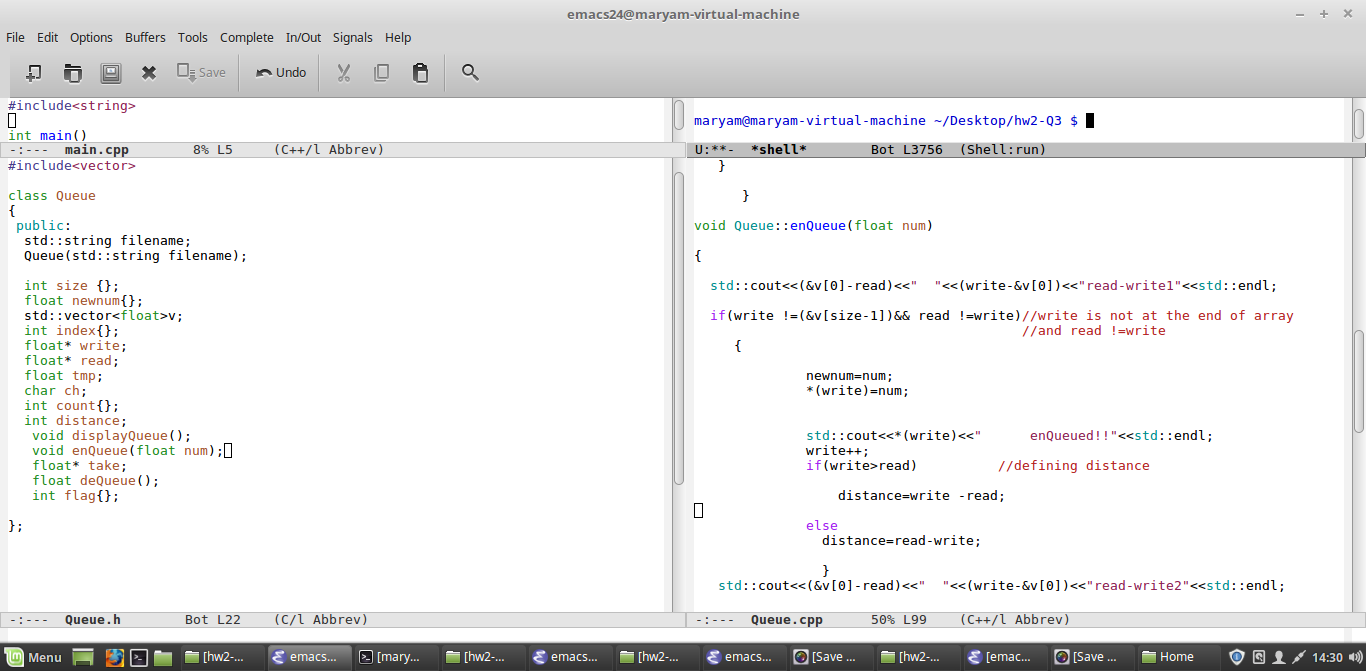
درصورتیکه read>write اعدادی را که در فاصله ی write تا read نیستند را چاپ میکنیم.

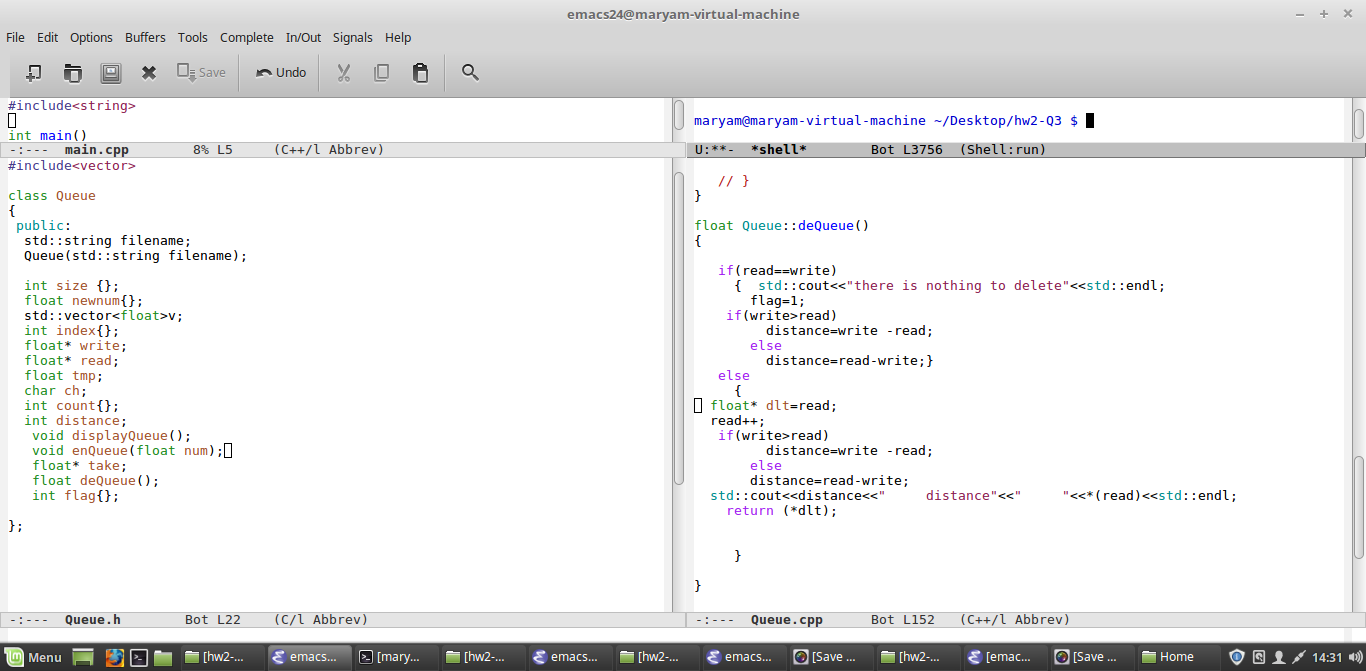






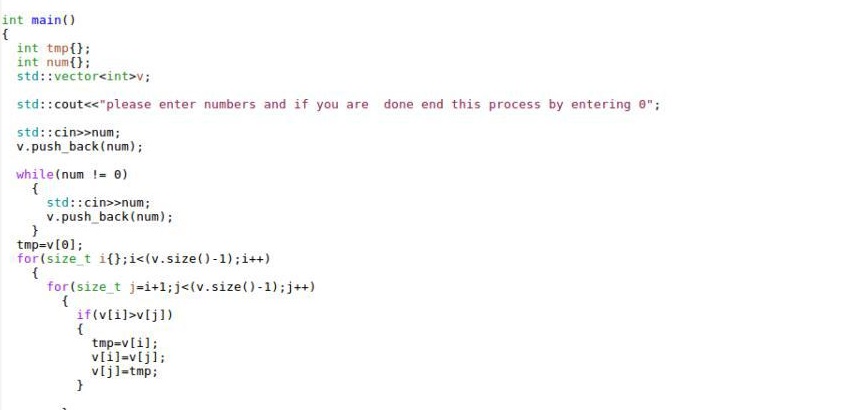


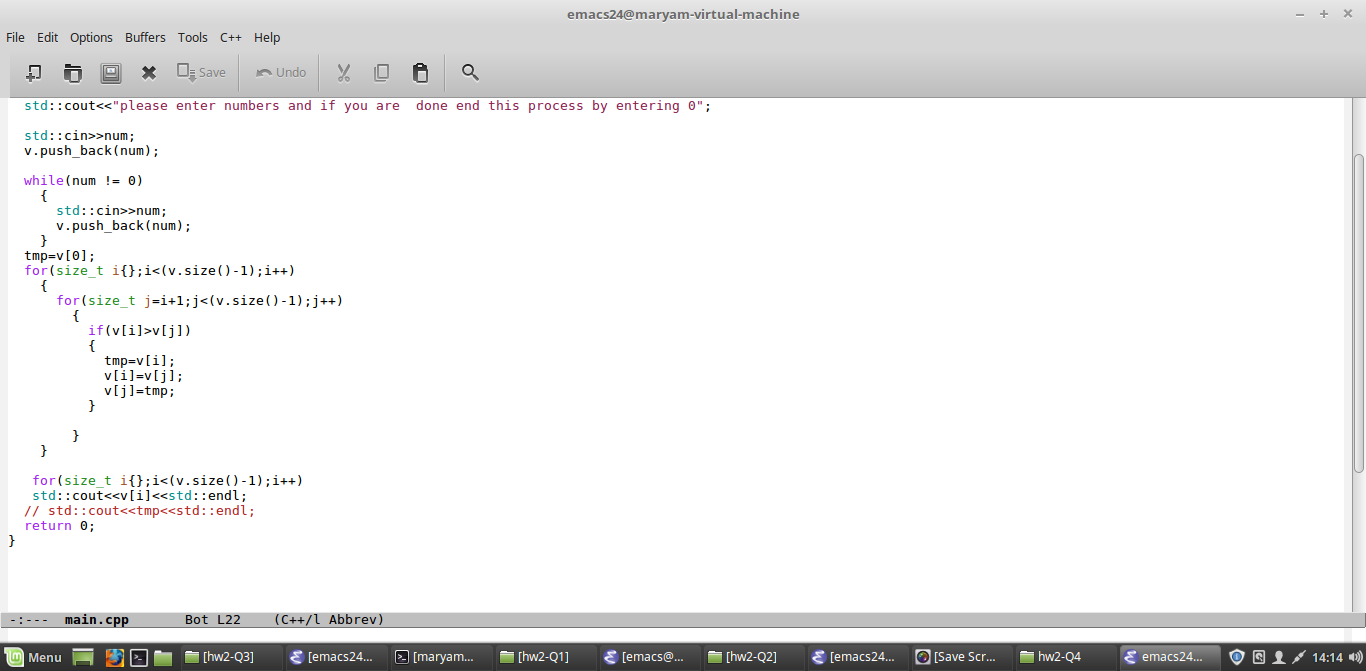




**سوال 4:**

**کد سوال:**

****



در این سوال میخواهیم از ورودی اعدادی را دریافت کنیم و این پروسه را با زدن عدد صفر توسط کاربر متوقف کنیم .

برای اینکار از حلقه ی while استفاده میکنیم و تا وقتی که متغیرمان برابر 0 نشده است از ورودی عدد دریافت میکنیم .

این اعداد را با استفاده از vector در یک آرایه ذخیره میکنیم

برای مرتب سازی این اعداد با استفاده از دو حلقه ی for هر بار در بین عناصر عنطر کوچکتر را پیدا میکنیم .

به این صورت که for بیرونی هر بار یک خانه از آرایه را در نظر میگیرد وfor درونی این المان را با بقیه ی المانهای آرایه مقایسه میکند

در صورتیکه این المان بزرگتر از المانی بود که با آن مقایسه میشود این خانه از آرایه را برابر مقدار المان کوچکتر قرار میدهیم

**منطق این الگوریتم به شرح زیر است:**

