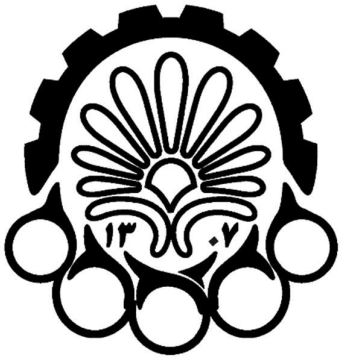

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

گزارش تمرین دوم

برنامه نویسی پیشرفته

دکتر جهانشاهی

پرهام کریمی ریکنده

9423090

تمرین اول :

 # کد شماره یک :

- **آیا کامپایل می شود؟**
  - بلی
- **برنامه نویس چه منطقی را دنبال می کرده است؟**
  - در واقع شاید عبارت const در تعریف b ما را به اشتباه بیندازد اما باید توجه کنیم که این تعریف متغیری به نام b تعریف میکند که از نوع اشاره گر ثابت است که این اشاره گر به یک متغیر از نوع int اشاره میکند. یعنی در ادامه ی برنامه، حق تغییر آدرس درون b را نداریم ولی می توانیم با استفاده از b، مقدار متغیری که به آن اشاره می کند را عوض کنیم.
- **خروجی برنامه؟**
  - 12 0xffffffff 12
  - که در آن، 0xffffffff مثلا آدرسی است که سیستم عامل به a اختصاص داده و اکنون در b ذخیره شده است.

 # کد شماره دو :

- **آیا کامپایل می شود؟**
  - خیر
  - از آنجایی که اشاره گر e به صورت ثابت تعریف شده است، در نتیجه در خط 10 که سعی در تعویض محتوای آن شده است، عمل کامپایل با خطا مواجه می گردد.
- **برنامه نویس چه منطقی را دنبال می کرده است؟**
  - دو متغیر b و c و یک ثابت a را در ابتدای برنامه تعریف نموده. در ادامه اشاره گر b را تعریف نموده که اشاره گری متغیر است که به const int اشاره می نماید؛ مقدار اولیه ای که به آن اختصاص داده شده نیز از همین نوع است؛ اما در ادامه، مقدار آن عوض شده و آدرس متغیر c که از نوع int است در آن ریخته شده؛ این عمل در واقع یک cast بوده که به صورت خودکار توسط کامپایلر انجام شده است.(لازم به ذکر است که اگر در ادامه بخواهیم مقدار c را توسط اشاره گر b عوض کنیم، کامپایلر اجازه ی این کار را نمی دهد چرا که نوع اشاره گر b به صورت "اشاره به ثابت" تعیین شده است.)
  - اشاره گر e که از نوع "اشاره گر ثابت به متغیر int" می باشد را تعریف نموده و با مقدار اولیه ی "آدرس متغیر c" آن را مقدار دهی می نمائیم. اما در خط بعد (10) که قصد تعویض مقدار آن را داریم، کامپایلر پیغام خطا خواهد داد. چرا که این اشاره گر از نوع ثابت بوده و مقدار آن از مقدار اولیه قابل تغییر نیست!
- **خروجی برنامه؟**
  - از آنجایی که برنامه کامپایل نمی شود، در واقع خروجی برنامه به طور کامل معنا ندارد اما بخشی از کد که مشکلی ندارد و منجر به خروجی می شود (خط 8) را داریم:
    - 10 0xffffffff 20
  - که در آن، 0xffffffff مثلا آدرسی است که سیستم عامل به c اختصاص داده و اکنون در b ذخیره شده است.

 # کد شماره سه :

- **چه نوع متغیری؟**
  - **خط 3:** متغیر p1 یک اشاره گر از نوع "اشاره کننده به const char" می باشد. یعنی نمیتوانیم توسط p1 مقدار متغیری را که به آن اشاره می کند، تغییر دهیم. (ولی آدرس درون آن را می توانیم تغییر دهیم چون یه اشاره گر ثابت نیست.)
- **آیا درست هستند؟**

- **خط 5:** این کار همانند کاری است که در "کد شماره 2" انجام شده؛ توضیحات مربوط به تبدیل نوع اشاره گر در آن بخش داده شد. در اینجا نیز همین تبدیل را داریم در واقع از آنجایی که با این کار داریم اشاره گر را محدود تر می کنیم، مشکلی بوجود نمی آید. اما برعکس این کار که در خط 9 انجام شده، امکان پذیر نیست.
- **خط 8:** چون p1 از نوع "اشاره گر به کاراکتر ثابت" است، نمی توانیم توسط آن، مقدار متغیری را که دارد به آن اشاره می کند، عوض کنیم. در نتیجه کامپایلر نیز از این کد خطا می گیرد.
- **خط 9:** در این جا میخواهیم مقدار یک "اشاره گر به متغیر کاراکتر" را برابر مقدار یک "اشاره گر به کاراکتر ثابت" قرار دهیم؛ در واقع با این کار محدوده ی name بازتر خواهد بود! چراکه اکنون می توانیم متغیری که name به آن اشاره می کند و ما تا بحال نمی توانستیم آن را از طریق name تغییر دهیم (چون name از نوع "اشاره گر به کاراکتر ثابت" بود)، از طریق p2 که از نوع "اشاره گر به متغیر کاراکتر" است، تغییر دهیم! اما کامپایلر جلوی این کلک ما را می گیرد! و پیغام خطا خواهد داد.

- خروجی کد در صورت وجود؟

- اگر از دو خط انتهایی صرفنظر کنیم، کد **"احتمالا"** اجرا خواهد شد. چراکه در خط 6 سعی در خواندن مقادیری داریم که در اختیار برنامه ی ما نیستند و ممکن است در بخشی از حافظه قرار گرفته باشند که سیستم عامل اجازه ی دسترسی به آن را به برنامه ندهد و از ادامه ی اجرای برنامه جلوگیری نماید.
    - Ami
    - a@@
- که @ ممکن است هر چیزی باشد.(بسته به چیزی که در آن لحظه در آن مکان از حافظه قرار داشته باشد.)

 # کد شماره چهار :

- چه نوع متغیری؟

- **خط 1:** اشاره گر p1 از نوع "اشاره گر به int" ساخته شده و با مقدار اولیه ی "آدرس خانه ی اول آرایه ی 10 خانه ای که توسط دستور new به صورت پویا ساخته شده (فضا از سیستم عامل گرفته شده)" مقداردهی می گردد.
- **خط 2:** یک آرایه ی 10 خانه ای از اشاره گر ها از نوع "اشاره گر به int" به نام p2 ساخته شده.
- **خط 3: !**
- **خط 4: !**
- **خط 5: !**

1- تمرین دوم :

- تشریح:

- هدف از این کد ساخت برنامه ای است که یک نوع از غلط های املایی انگلیسی را تشخیص بدهد. برای این کار ابتدا یک استریم که شامل فایل متن مورد بررسی تعریف شد. برای بیشینه کردن تعداد کاراکتر های قابل بررسی، فضاهای خالی اضافی (white space ها) تماما حذف گردیده و با تنها یک فاصله جایگزین می گردند برای این کار یک استرینگ (رشته ی متنی) موقت نیز تعریف شد.
- رشته ی حروفی که در متن باید به دنبال آنها بگردیم، که همان حروف بی صدای کوچک هستند نیز تعریف گردید.
- بقیه ی متغیر ها، در جایگاه خودشان در صورت لزوم توضیح داده خواهند شد.
- **Importing the text to str_txt:**
    - در گام اول هر بخش از متن را که به یک فضای خالی (white space) منتهی می شود، به داخل رشته ی موقت ریخته و سپس آن رشته ی موقت را به همراه یک فاصله، به رشته ی اصلی اضافه می نمائیم و این عمل تا زمانی ادامه می یابد که استریم با مشکل مواجه

نشود و همچنین در صورت رسیدن به انتهای استریم، حلقه متوقف شده و همچنین، عمل اضافه کردن رشته ی موقت به رشته ی اصلی نیز انجام نمی گردد. در انتهای این بخش هم استریم بسته می شود.(زیرا اکنون تمام متن مورد بررسی در رشته ی اصلی داخل برنامه ذخیره شده است و دیگر نیازی به باز ماندن استریم نیست!)

- **Extracting non vowel letters' position:**
  - در ابتدای این بخش یک آرایه تعریف شده که به صورت پویا ساخته شده تا از نظر حجمی محدودیت بوجود نیاید و همچنین از استفاده از روش معمول پرهیز شود. (روش معمول در چنین برنامه هایی (البته در سطح آموزشی) این است که یک آرایه با طول بسیار زیاد ساخته شود.) با چنین تعریفی، تنها به تعداد حروف صدا دار و حروف بی صدای بزرگ و علائم موجود در متن، خانه ی اضافی موجود است! وظیفه ی این آرایه ذخیره ی مکان های مربوط به کاراکترهایی از متن مورد بررسی است که یکی از کاراکتر های موجود در رشته ی "حروف بی صدای کوچک" باشند.
  - حلقه ی جستجو تا ادامه خواهد یافت تا زمانی که مقدار offset از طول رشته ی اصلی منهای 1، کمتر باشد. چراکه میدانیم آخرین کاراکتر حتما فاصله خواهد بود و نیازی به بررسی آن نیست!
- **Detect and show the mistakes:**
  - در این بخش از زاویه ای دیگر به موضوع نگاه شده! بررسی به صورت یک خانه به جلو در هر بار دور زدن حلقه انجام می شود که مانند بررسی یک موج مربعی می باشد! به این صورت که موج مربعی به طور منظم با هر کلاک (هر بار دور زدن حلقه) بررسی می گردد و طول 1 بودن آن (توالی حروف کوچک) بررسی می گردد و در صورت بیشتر یا مساوی 5 بودن آن، اعلان خطا می گردد و خطای مذکور نمایش داده می شود.
  - همانطور که بیان شد این موضوع **"مانند"** بررسی یک موج مربع است چراکه در این برنامه از نگاه به آینده استفاده شد که در واقع غیر علی بوده و پیاده سازی آن فقط در صورتی امکان پذیر است مانند همینجا، داده ی ما ذخیره شده باشد و ما قصد بازخوانی آن را داشته باشیم.
  - برای اجرای این الگوریتم از یک پرچم کمک گرفته شده که در صورت یافتن توالی یک می گردد.
- **Delete Dynamic Variables:**
  - از آنجایی که در این برنامه از متغیر پویا استفاده شد، باید آن را در انتهای برنامه حذف نمائیم.
  - البته این عمل به صورت خودکار توسط سیستم عامل در انتهای اجرای برنامه صورت می گیرد اما باید توجه داشت که ممکن است این برنامه، بخشی از یک برنامه ی بزرگ تر باشد که در این صورت در صورت پاک نکردن آن، تا انتهای آن برنامه باقی خواهد ماند!

- **نکات به کار گرفته شده در این کد:**

a. با توجه به نوع متغیر هایی که جهت آدرس دهی ها در این برنامه استفاده شده، تعداد کاراکترهای متن مورد بررسی نمی تواند از $4294967295 = 1 - 2^{32}$ بیشتر باشد.

b. در تعریف آرایه ی txt_2_nmbr حتما باید مقادیر اولیه صفر باشند زیرا در ادامه ی برنامه از این ویژگی استفاده شده است. (این کار نیز توسط لیست مقداردهی اولیه انجام شده که از ویژگی های جدید این زبان می باشد (C++11))

2- تمرین سوم :

- **تشریح:**
  - هدف در این سوال پیاده سازی یک صف حلقوی است؛ برای این کار کلاسی که از روی یک فایل ساخته می شود، داریم و این کلاس متغیر های اصلی ذیل را دارد:
    - سایز و محتوا
    - مکان نوشتن و مکان خواندن و پرچم لبریز یا خالی بودن
  - **Queue::Queue (const std::string& file_add); :**

- در ورودی کانستراکتور یک رشته ی متنی (به صورت رفرنس) گرفته می شود که این رشته ی متنی در واقع آدرس دسترسی به فایل مقداردهی اولیه می باشد. (در صورتی که فایل در پوشه ی برنامه موجود باشد، تنها نام فایل به همراه پسوند کافی خواهد بود)
- در گام اول سایز (اندازه) ی صف را از فایل استخراج میکنیم و پس از ساختن آرایه ای پویا به اندازه ی سایز استخراج شده، به استخراج و قرار دادن مقادیر صف در درون آرایه پرداخته شده است. این کار تا جایی انجام می گردد که یا به انتهای صف رسیده باشیم و یا مقادیر تمام شده باشند (به انتهای فایل رسیده باشیم).
- در انتها نیز تمامی خانه های خالی آرایه (صف) را با nan (Not A Number) پر کردیم. از آنجایی که نمیدانستم چطور به صورت مستقیم این مقدار را بدهم، از خروجی تابع std::stof استفاده کردم. در ادامه نیز هر گاه خانه ای از آرایه (صف) خالی باشد، با این مقدار مقداردهی می گردد.
- در تعیین مکان خواندن از آنجایی که در این بخش داریم برای اولین بار صف را مقدار دهی می کنیم، پس حتما میدانیم که طبق برنامه ی نوشته شده، اولین خانه ی نوشته شده و در نتیجه اولین خانه برای خوانده شدن، خانه ی 0 است.
- در تعیین مکان نوشتن:
  - اگر صف لبریز شده باشد، مکان نوشتن را در 0 قرار می دهیم
  - در باقی حالات مکان نوشتن را در یک خانه (با توجه به حلقوی بودن صف) جلوتر از آخرین خانه ی نوشته شده قرار دادیم.

- **: void Queue::displayQueue () const;**
  - در وضعیت خالی پیغام مربوطه نمایش داده می شود
  - در وضعیت لبریز هم به ترتیب از مکان خواندن تا انتها و از ابتدا تا قبل مکان خواندن نمایش داده می شود که این نوع ترتیب به دلیل حلقوی بودن صف می باشد.(اولویت نمایش با اولویت ورود به صف می باشد.)
  - در وضعیت عادی از مکان خواندن تا قبل از مکان نوشتن نمایش داده می شود.
  - در وضعیت منقطع نیز مانند وضعیت لبریز عمل می شود منتها این بار تا قبل مکان نوشتن نمایش داده می شود. (در واقع می توانیم این دو وضعیت را در یکدیگر ادغام نمائیم و از نوشتن این دو، به صورت جداگانه خودداری نمائیم)
- **: float Queue::deQueue();**
  - در نوشتن این تابع نیز مانند تابع قبلی عمل شده + توجه و عمل متناسب با موارد زیر:
    - **1-** برای خواندن مقدار در مکان خواندن، از یک متغیر موقت استفاده شده
    - **2-** برای پاک کردن مقدار مورد نظر nan در آن خانه نوشته شد
    - **3-** پس از خواندن مقدار و پاک کردن خانه در مکان خواندن، مکان خواندن به یک خانه جلوتر (با توجه به حلقوی بودن صف) منتقل می گردد.
    - **4-** قبل از هر دستور برگرداندن مقدار (return) بررسی شده که آیا مکان خواندن با مکان نوشتن برابر شده یا نه که از آنجایی که در این تابع، عمل حذف صورت میگیرد نه نوشتن**!**، پس در صورت رخ دادن چنین وضعیتی، یعنی صف کاملا خالی شده است.
    - **5-** در انتها نیز مقدار nan برگردانده شده که با توجه به کد، یعنی یا خطایی صورت گرفته و یا صف کاملا خالی بوده که تا به این جای برنامه، چیزی برگردانده نشده**!**
- **: void Queue::enQueue (float in_num);**
  - در این تابع نیز در صورتی که صف کاملا پر باشد، پیغام مربوطه نمایش داده می شود.
  - در صورت لبریز نبودن صف، مقداری که به صورت ورودی به تابع داده شده، در مکان نوشتن نوشته می شود و مکان نوشتن به یک خانه جلوتر (با توجه به حلقوی بودن صف) منتقل می گردد.
  - در انتها نیز بررسی می گردد که آیا مکان خواندن و مکان نوشتن برابر شده است یا خیر، که در صورت برابر شدن، به معنای لبریز شدن صف (زیرا با انجام عمل نوشتن، مطمئنا صف "کاملا خالی" نشده**!**) می باشد. (پرچم لبریز بودن مقدار دهی می شود.)

- **نکات به کار گرفته شده در این کد:**
  - پرچم لبریز یا خالی بودن تنها در مواقعی اهمیت پیدا می کند که صف کاملا خالی (خالی) یا کاملا پر (لبریز) باشد؛ نشانه ی چنین وضعیتی نیز، برابر شدن مقادیر مکان نوشتن و مکان خواندن می باشد. در واقع این پرچم برای تفکیک دو حالت ممکن در هنگام برابر شدن این دو مقدار در نظر گرفته شده است.
  - به طور کلی در بخش های مختلف در هنگام نوشتن کد، برای صف، 4 وضعیت کلی در نظر گرفته شد:
    1. کاملا خالی
    2. کاملا پر (لبریز)
    3. حالت عادی : در این وضعیت از نظر شماره ی خانه ی آرایه ای که اعضای صف در آن قرار دارند، پیوستگی وجود دارد.
    4. حالت منقطع : در این وضعیت از نظر شماره ی خانه ی آرایه ای که اعضای صف در آن قرار دارند، نا پیوستگی وجود دارد؛ در واقع بخشی از صف در ابتدا و بخش دیگر آن در انتهای آرایه قرار گرفته است.

3- تمرین چهارم :

- **تشریح:**
  - هدف در این سوال مرتب سازی به روش انتخابی است
  - **Definition of variables :**
    - از آنجایی که تعداد اعداد وارد شده معلوم نیست، از بردار (vector) استفاده نمودیم تا طول آن متغیر باشد. و از آنجایی که نوع اعداد نیز معلوم نیستند، از نوع float تعریف نمودیم تا رنج بیشتری از اعداد را شامل شود.
  - **Getting numbers :**
    - در این بخش مقادیر وارد شده توسط کاربر را در بردار ذخیره می نمائیم و تا زمانی این اعداد را از کاربر میگیریم که کاربر عدد 0 را وارد کند. (در صورتی که کاربر همه را پشت سر هم با فاصله نوشته باشد و در نهایت enter را بزند (\n را وارد کند) نیز تا قبل عدد 0 مقادیر خوانده و ذخیره می گردند.)
  - **Show the scrambled numbers that user had entered! :**
    - یک بار اعداد وارد شده توسط کاربر را به صورت خام (مرتب نشده) نمایش می دهیم.
  - **Sorting the numbers with "Selection Sort" method :**
    - در هر بار عمل انتخاب، بر روی خانه ای از بردار قرار میگیریم، سپس تک تک خانه های بعد از آن تا انتهای بردار را بررسی می کنیم، در صورتی که مقدار خانه ای، کوچکتر از مقدار کمینه (min) موقت ما بود (که اولین مقدار آن، مقدار خانه ای است که در آن قرار داریم) مقدار و شماره ی آن خانه را در متغیر موقت ذخیره می کنیم و در انتها مقدار خانه ی شامل مقدار کمینه ی بدست آمده را با مقدار خانه ای که در آن قرار داریم، عوض می کنیم. (در صورتی که خانه ای که در آن قرار داریم، خودش کمینه باشد، در این صورت، عمل تعویض انجام نمی شود.)
  - **Show the sorted numbers :**
    - اعداد مرتب شده را نمایش می دهیم.
- **نکات به کار گرفته شده در این کد:**
  - در توضیحات هر بخش ذکر شد.

4- Git :

- از آنجایی که قبلا در سایت github ثبت نام کرده بودم، مراحل ثبت نام را دوباره طی نکردم و این تمرین را در قالب همین سایت انجام دادم:
- پس از ساختن پروژه با نام گفته شده، همانطور که در راهنمائی خود سایت نیز گفته شده عمل کردم و در ادامه نیز با استفاده از نرم افزاری که در سایت موجود است به صورت گرافیکی اعمال مورد نیاز را انجام دادم.

- برای فایل gitignore. باید فرمت ها و هم چنین فایل هایی که نمیخواهیم توسط git دنبال شوند، در آن ذکر کنیم (ممکن است از بین فرمت های مجاز، بخواهیم چند فایل خاص نیز دنبال نشوند که در چنین شرایطی خود آن فایل ها را نیز در اینجا باید ذکر کنیم.) برای این کار از یکی از نمونه های همین سایت استفاده کردم که در این لینک موجود است:
    - **https://github.com/github/gitignore/blob/master/C%2B%2B.gitignore**