مدرسان: **رامتین خسروی، محمدامین صادقی** طراحان: **آبتین باطنی، شایان پاکزاد، یاسمن جعفری**

مهلت تحویل: دوشنبه ۲۱ام اسفند ۱۳۹۶ ، ساعت ۲۳:۵۵

مهندس کامپیوتر نمونه!

فرض کنید به دلیل افزایش قیمت ترافیک اینترنت از شما می‌خواهند که مکانیزمی را ایجاد کنید که فایل‌های رد و بدل شده با حجم کمتری رد و بدل شوند در نتیجه این درخواست شما به عنوان یک مهندس کامپیوتر با هوشمندی تمام! تصمیم می‌گیرید از یک روش فشرده‌سازی برگشت‌پذیر استفاده کنید، سپس به شما اطلاع می‌دهند که جاسوسانی وجود دارند که فایل های رد و بدل شده بین اعضای شرکت را دریافت و با انجام تغییراتی در محتوای آن‌ها، کارها و دستورات مدنظر خودشان را جایگزین کارها و دستوراتی که واقعی می‌کنند. برای همین آن‌ها از شما درخواست روشی برای حل این مشکل را دارند. در اینجا شما دوباره هوشمندی به خرج می‌دهید! و برای اینکه جلوی این کار زشت جاسوسان را بگیرید پیشنهاد یک روش رمزنگاری متقارن برروی فایل‌های فشرده شده را می‌دهید.

فشرده سازی و رمزنگاری

برنامه‌ی شما باید چهار عمل فشرده‌سازی، بازسازی اطلاعات فشرده‌شده، رمزنگاری و رمزگشایی را روی فایل‌هایی که به عنوان ورودی به آن داده می‌شود را انجام دهد و یک فایل خروجی تولید کند.

پیاده‌سازی این تمرین، شامل دو بخش اصلی فشرده‌سازی و رمزنگاری است. شما باید از روش کدگذاری هافمن[1] برای فشرده سازی و تکنیک سزار[2] برای رمزنگاری، استفاده کنید.

کدگذاری هافمن، یک روش فشرده سازی بدون اتلاف اطلاعات می باشد. مبنای کار این الگوریتم، اختصاص دادن کدهایی با طول متغیر، به کاراکترهای ورودی است، به طوری که طول کدها وابسته به تعداد تکرار هر کاراکتر باشد. به عبارتی، هرچقدر تعداد تکرار کاراکتری بیشتر باشد، طول کد مربوط به آن کمتر خواهد بود.

رمزنگاری سزار، یکی از ساده ترین و شناخته شده ترین روش های رمزنگاری است که در آن هر کاراکتر از متن اصلی، با کاراکتری دیگر با فاصله ثابت k جایگزین می شود. این روش، با وجود اینکه به راحتی قابل شکسته شدن است، یکی از اولین روش های رمزنگاری بوده که در زمان خود، روشی امن به حساب می آمد.

هدف این تمرین افزایش توانایی شما در طراحی بالا به پایین مساله و شکستن پروژه به مساله‌های کوچک‌تر است، پس باید بتوانید به شیوه‌ی مناسب از توابع استفاده کنید و به‌درستی مساله را در به تابع‌های کوچک‌تر افراز کنید. برای همین **بخش‌زیادی از نمره‌ی این تمرین وابسته طراحی درست برنامه شما می‌باشد.**

پیاده سازی

شما در این پروژه باید عملیات زیر را پیاده سازی کنید. برای راحتی شما پیشنهاد می شود که این مراحل را با ترتیب ذکر شده انجام دهید.

خواندن از فایل

در این بخش، شما باید فایل ورودی را بایت به بایت خوانده و تعداد تکرار هر یک از کاراکترها را بدست آورید. این اطلاعات در کدگذاری هافمن، استفاده خواهد شد.

[1] Huffman Coding

[2] Caesar

ساخت درخت هافمن

برای پیاده سازی الگوریتم هافمن، از ساختار داده[3] درخت استفاده می شود. هر برگ از این درخت، یکی از کاراکترها می باشد. در هر مرحله، دو گره با کمترین تعداد تکرار ادغام شده و گره جدیدی با تعداد تکرار مجموع تکرارهای دو گره ترکیب شده، ایجاد می‌کنند. یکی از گره های قبلی فرزند راست و دیگری فرزند چپ گره جدید خواهد بود. این کار را تا زمانی که تنها یک گره باقی بماند، ادامه می دهیم. توجه داشته باشید که درخت ذکر شده، دودویی[4] می باشد.

در آخر، کافی است به هر یک از یال های راست و چپ گره ها، یکی از اعداد ۰ یا ۱ را نسبت دهیم. کد معادل هر کاراکتر، برابر با رشته ارقام دیده شده در مسیر بین ریشه‌ی درخت تا برگ متناظر آن کاراکتر، خواهد بود.

در این قسمت، شما باید درخت هافمن را تشکیل داده و با حرکت روی درخت، کد متناظر هر کاراکتر را بدست آورید و سپس این اطلاعات را در ترمینال چاپ کنید.

فشرده سازی

با استفاده از کدهای بدست آمده از درخت هافمن، هر یک از کاراکترهای فایل ورودی را با کد مربوطه آن، جایگزین کرده و داده‌های نهایی‌تان را در یک string ذخیره کنید. در این مرحله برای اینکه بعدا توانایی بازیابی داده‌های اولیه را داشته باشید شما **باید** کدهای متناظر به هر کاراکتر را نیز در ابتدای این رشته به فرمت روبرو ذخیره کنید. در خط اول ۲۵۶ رشته از کاراکترهای صفر و یک قرار دهید که رشته i ام کد مربوط به کاراکتر i ام بدست آمده توسط درخت هافمن می‌باشد. در انتهای این خط و پس از آخرین رشته تعداد **بیت های** فایل فشرده شده را قرار دهید. در خط بعدی حاصل فشرده سازی قرار می‌گیرد که مجموعه‌ای از کاراکترهای به ظاهر نامرتبط و بی‌ربط است. به طور مثال فایل خروجی‌تان می‌تواند به شکل زیر باشد:

```
01101 0001 10 0001 1111 … 1110 24
#i^
```

در خط اول فایل خروجی رشته‌ی باینری توصیه شده توسط الگوریتم هافمن را برای هر کاراکتر داریم به همراه طول کلی پیام که عدد 24 می‌باشد. خط بعدی از سه کاراکتر تشکیل شده اگر رشته‌ی بیتی متناظر به هر کدام را در c++ در نظر بگیریم به یک رشته‌ی 24 بیتی می‌رسیم که سپس با توجه به خط دوم باید آن را رمزگشایی کنیم و پیام اصلی را از آن استخراج کنیم. اگر در عوض عدد 24 مثلا عدد 23 قرار داشت آنگاه شما باید این سه کاراکتر را به یک رشته‌ی بیتی تبدیل کنید ولی بیت نهایی و آخر آن را حذف کنید تا یک پیام به طول 23 داشته باشید.

رمزنگاری

داده‌ی فشرده شده را کاراکتر به کاراکتر خوانده و با الگوریتم سزار را با کلید مشخص شده (که به عنوان ورودی به برنامه‌ی شما داده می‌شود) روی آن، اجرا کنید.

درهم سازی[5]

با رد و بدل شدن فایل‌ها بر بستر شبکه و اینترنت احتمال از بین رفتن یا تغییر در بیت‌های فایل وجود دارد.

برای این که مشخص شود فایلی که رد و بدل شده نسبت به فایل اصلی تغییری نکرده است از توابع درهم سازی استفاده می‌شود. این توابع یک رشته با اندازه مشخص به فایل اختصاص می‌دهند که با انجام یک الگوریتم درهم‌سازی روی تمام بیت‌های فایل به دست می‌آید. پس از آن هر بار همراه فایل این رشته هم ارسال می‌شود و گیرنده با اجرای دوباره الگوریتم روی فایل دریافت شده این رشته را به دست می‌آورد و با رشته‌ی دریافت شده مقایسه می‌کند. حال اگر بیتی‌ای در فایل تغییر کرده باشد این دو رشته یکسان نمی‌شوند و می‌توان متوجه شد که فایل دریافت شده معتبر نیست. الگوریتم‌های بسیاری برای تخصیص یک رشته به فایل‌ها ایجاد

[3] Data Structure

[4] Binary

[5] Hashing

شده‌اند. پیشنهاد می‌شود برای تفریح(!) و آشنایی بیشتر از دستور md5sum در ترمینال لینوکس استفاده کنید تا با نحوه‌ی عملکرد این الگوریتم‌ها آشنایی اولیه پیدا کنید.

الگوریتم درهم سازی SDBM

۱. ابتدا متغیر از نوع long long به نام hash را برابر صفر قرار بده.
۲. به ازای هر یک کاراکتر در خروجی متغیر hash را در ۲۶ و همچین $۲^{۱۶}$ برابر خود ضرب کن و مجموع این دو عدد را در یک متغییر ثانویه بریز و سپس مقدار اولیه آنرا از عدد ثانویه کم کن. در انتها عدد ثانویه را با کاراکتر فعلی جمع کن به عنوان مقدار جدید hash قرار بده.
۳. پس از اجرای ۲ برای تمام کاراکترها کار تمام است و hash به مقدار نهایی خود رسیده است.
برنامه‌ی شما باید پس از فشرده‌سازی و رمزنگاری فایل، با استفاده از تابع درهم‌سازی، رشته حاصل از اجرای الگوریتم بالا روی داده‌ها را به‌دست آورد و یک عدد از نوع Long Long در خط اول خروجی قرار دهد. سپس بقیه داده‌ها را از خط دوم به بعد در فایل خروجی قرار دهد.

بازیابی فایل اولیه

حال شما باید بتوانید فایل نهایی(فشرده و رمزگذاری شده) را رمزگشایی کنید. فایل به‌دست آمده را به صورت جداگانه، ذخیره کنید. در این مرحله باید تابع درهم‌سازی را روی داده‌های فایل (خط دوم به بعد) اجرا کنید و مقدار حاصل را با خط اول فایل مقایسه کنید و در صورت وجود تناقض با یک پیام مناسب به کاربر اعلام کنید.

نحوه اجرای برنامه

برنامه‌ی شما باید بتواند توسط آرگومان‌هایی که به عنوان ورودی به آن داده می‌شود (به ترتیب نام فایل وردی، نام فایل خروجی و کلید رمزنگاری) به یکی از دو حالت زیر اجرا شود. (شیوه‌ی تشخیص و استفاده از آرگومان‌ها را از اینجا بخوانید.)

```
a.out encode <inputFileName> <outputFileName> <kNumber>
a.out decode <inputFileName> <outputFileName> <kNumber>
```

برای اجرای دستور دوم مراحل گفته شده را به صورت معکوس اعمال کنید و فایل نهایی را ذخیره کنید.
علاوه بر این برنامه‌ی شما باید از دستورات زیر نیز پشتیبانی کند

```
a.out compress_only <inputFileName> <outputFileName>
a.out decompress_only <inputFileName> <outputFileName>
```

در دستور اول باید تا مرحله‌ی الگوریتم هافمن را بروی inputFileName اجرا کنید و سپس نتیجه را در outputFileName ذخیره کنید. (اگر بخش امتیازی را نیز پیاده سازی کرده‌اید بعد از هافمن الگوریتم دوم را اجرا کنید.) در دستور دوم باید فایل ورودی را از حالت فشرده خارج کنید.

مرحله‌ی آزمودن

از آنجایی که آزمودن کد یکی از مراحل بسیار پراهمیت در تولید نرم افزار به شمار می‌آید، سعی کنید **در هنگام پیاده سازی همواره از توابع کوتاه و کاربردی استفاده کنید**، همچنین شما باید به انتخاب خودتان برای ۱۰ عدد از توابعی که در برنامه‌ی خود پیاده‌سازی کرده‌اید، یک تابع آزمون بنویسید که درستی کارکرد تابع موردنظر را بررسی کند.

بخش امتیازی

روش LZ77 هم یکی از روش‌های پایه‌ی فشرده سازی است که به همراه روش هافمن برای تولید فایل‌های زیپ استفاده می‌شود. با استفاده از یک تابع بازگشتی این روش را پیاده سازی کنید و بر روی داده‌های فایل اعمال کنید. برای اطلاع بیشتر درباره‌ی این روش لینک‌های قرار داده شده را مطالعه کنید.

نحوه‌ی تحویل

فایل برنامه‌ی خود را با نام A3-SID.zip در صفحه‌ی CECM درس بارگذاری کنید. برای مثال، اگر شماره‌ی دانشجویی شما ۸۱۰۱۱۲۳۴۵ است، نام فایل شما باید A3-810112345.zip باشد.

لطفاً از روش‌های دیگر فشرده‌سازی مانند rar یا tar.gz استفاده نکنید.

- برنامه‌ی شما باید در سیستم‌عامل لینوکس و با مترجم g++ ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- به فرمت و نام فایل‌های خود دقت کنید.
- از صحت فرمت ورودی‌ها و خروجی‌های برنامه‌ی خود مطمئن شوید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

مراجع

- الگوریتم سزار:

https://learncryptography.com/classical-encryption/caesar-cipher

- الگوریتم هافمن:

https://www.siggraph.org/education/materials/HyperGraph/video/mpeg/mpegfaq/huffman_tutorial.html

- الگوریتم LZ77:

https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Text/02-string_encoding/01-LZ77/index.html