به نام خدا

داده ساختارها و الگوریتم‌ها
Data Structures & Algorithms

دانشگاه شهید باهنر کرمان
بخش مهندسی کامپیوتر
مدرس: حدیث محسنی

تمرین برنامه نویسی ـ سری اول **موعد تحویل: دوشنبه، ۲۰ فروردین ۹۶**

توجه ۱: انجام تمرین‌ها به صورت گروهی قابل قبول نیست و به افرادی که تمرین‌های مشابه تحویل دهند، **هیچ** نمره‌ای تعلق نخواهد گرفت. مسلماً همفکری اشکالی ندارد، اما همه فرق همفکری و تقلب (کپ زدن) را می‌دانیم!
توجه ۲: به موعد تحویل دقت فرمایید. **به هیچ بهانه‌ای مثل تعطیلات عید و خبر نداشتن از تمرین‌ها و ... مهلت تمدید نخواهد شد.**
توجه ۳: فایل نهایی خود را به آدرس hmcourse@gmail.com بفرستید و subject آن را DS1 قرار دهید. به فایل‌هایی که به صورت دیگری ارسال شده باشند، نمره‌ای تعلق نمی‌گیرد. قبل از ارسال فایل نهایی خود، حتماً توضیحاتی که در ادامه آمده است را به دقت بخوانید.

برای اینکه درک واقعی و عملی از روش‌هایی که می‌خوانید پیدا کنید، کد مسائل زیر را به هر زبانی از خانواده C که دوست دارید (مثل C++ ، C#) پیاده سازی کنید. فایل‌های مربوط به هر سوال را در یک folder که نام آن همان شماره‌ی سوال است، قرار داده، سپس دو folder را در یک فایل zip یا rar به شماره دانشجویی خودتان قرار داده و فقط همین فایل نهایی را ارسال کنید. توجه داشته باشید که فقط فایل‌های مربوط به کدها (همان‌هایی که خودتان نوشتید! و نه فایل‌هایی که در زمان کامپایل و اجرا تولید شده‌اند) را ارسال کنید و از ارسال فایل‌های دیگر (مانند فایل exe و ...) خودداری کنید تا در ارسال فایل‌ها به مشکل نخورید.

مسئله‌ی ۱ : شمارش تعداد اجرای خطوط برنامه

شبه کد بازگشتی زیر را در نظر بگیرید و با پیاده‌سازی آن تعداد اجرای دستور print را به ازای ورودی‌های مختلف بشمارید. انتظار دارید چند خط در خروجی چاپ شود؟ آیا تعداد خطوط چاپ شده در خروجی با انتظار شما یکسان است؟

```
function Concow (n)
if n ≤ 1 then
    Print "STOP!'
    return 1
else
    for i ← 1 to n do
        Print Concow(⌊x/2⌋ )
    return n
```

مسئله ۲. مرتب سازی و مرتبه‌ی اجرایی

در این تمرین هدف درک زمان اجرا و مرتبه‌ی اجرایی الگوریتم‌های مختلف است. به این منظور، دو الگوریتم مرتب‌سازی معروف!، یعنی مرتب‌سازی درجی (InsertionSort) و مرتب‌سازی ادغامی (MergeSort) را باید با هم مقایسه کنید.

الگوریتم‌های فوق را به صورت دو تابع جداگانه (توابع با نام‌های InsertionSort و MergeSort) بنویسید که ورودی هر تابع یک آرایه از اعداد است که قرار است مرتب شود. خروجی هر تابع هم شامل دو عدد صحیح است که اولی نشان‌دهنده تعداد مقایسه‌های انجام شده و دومی نشان دهنده‌ی تعداد جابه‌جایی‌های انجام شده در الگوریتم مرتب‌سازی است.

بعد از پیاده‌سازی دو تابع مرتب‌ساز، حالا یک تابع دیگر به نام Comparison بنویسید که ورودی آن عدد صحیح n یا همان سایز آرایه‌ی اعداد است (تابع خروجی ندارد). این تابع، یک آرایه با سایز فوق را ساخته و با اعداد تصادفی صحیح مثبت پر می‌کند. از این آرایه به عنوان ورودی هر دو تابع مرتب‌ساز استفاده می‌شود. در این تابع، با صدازدن دو تابع مرتب‌ساز با آرایه‌ی ساخته شده، خروجی دو تابع مرتب‌ساز با هم مقایسه می‌شود تا تعیین شود که کدام روش مرتب‌سازی سریعتر عمل کرده است. هم‌چنین تعداد مقایسه‌های هر روش مرتب‌سازی با مرتبه‌ی اجرایی مورد انتظار از آن مقایسه می‌شود. نتایج این مقایسه‌ها باید به صورت جملات مناسب در خروجی چاپ شود.

برای درک کامل از عملکرد روش‌های مرتب‌سازی، تابع Comparison را به ازای $n = ۱۰, ۱۰۰, ۱۰۰۰, ۵۰۰۰$ اجرا کنید، خروجی‌ها را چاپ کنید و نتایج به دست آمده را با انتظاراتی که از این روش‌های مرتب‌سازی دارید، برای خودتان تحلیل کنید.

توجه داشته باشید که نیازی به چاپ آرایه‌ی ورودی یا مرتب شده نیست.

با آرزوی موفقیت! ☺