بسمه تعالی

دانشکده مهندسی برق و کامپیوتر

برنامه‌سازی پیشرفته

تمرین‌های کوتاه

توابع بازگشتی

۱. مدیریت حافظه در توابع

تمرین ۱ – برنامه‌ی کوچکی بنویسید که تفاوت زمان اجرای دو تابع یکسان که پارامتر یکی با مقدار و دیگری با ارجاع رد شده باشد را نشان دهد. برای این کار باید به کمک حلقه‌ها دسترسی به پارامترها را به تعداد زیاد انجام دهید.

تمرین ۲ – این دو نسخه از تابع مرتب‌سازی حبابی را در نظر بگیرید:

```
void bubble_sort_1(vector<int>& v) {
    for (int i = 0; i < v.size()-1; i++)
        for (int j = 0; j < v.size()-i-1; j++)
            if (v[j] > v[j+1])
                swap(v[j], v[j+1]);
}

void bubble_sort_2(vector<int>& v) {
    vector<int> w = v;
    for (int i = 0; i < w.size()-1; i++)
        for (int j = 0; j < w.size()-i-1; j++)
            if (w[j] > w[j+1])
                swap(w[j], w[j+1]);
    v = w;
}
```

تفاوت این دو نسخه در این است که در نسخه دوم، بردار پارامتر v در یک متغیر محلی به نام w ذخیره می‌شود، عملیات مرتب‌سازی روی آن انجام می‌شود و نتیجه در پارامتر v کپی می‌شود. به نظر شما کدام‌یک از این دو تابع سریع‌تر عمل می‌کنند؟ با اجرای این دو تابع روی بردارهای بزرگ زمان اجرای آنها را با هم مقایسه کنید و نتیجه را تفسیر نمایید.

۲. توابع بازگشتی

تمرین ۳ – تابع power(x, n) مقدار x^n را محاسبه می‌کند. این تابع را با فرض نامنفی‌بودن n به طور بازگشتی بنویسید. اگر مقادیر منفی را نیز برای n مجاز بدانیم این تابع را بازنویسی کنید.

تمرین ۴ – تابع binary(n) نمایش عدد n را در مبنای ۲ می‌نویسد. این تابع را به صورت بازگشتی بنویسید.

تمرین ۵ – یک تابع «غیربازگشتی» بنویسید که با دریافت یک عدد صحیح، مجموع ارقام آن را برگرداند (مثلاً با دریافت ۳۵۱ مقدار ۹ را برگرداند). این تابع را به صورت «بازگشتی» هم بنویسید. در نوشتن این تابع از بردار یا رشته استفاده نکنید.

تمرین ۶* – نظیر سؤال قبل تابعی بازگشتی بنویسید که یک عدد را دریافت کرده ارقام آن را برعکس کند و برگرداند. مثلاً با دریافت ۳۵۱، عدد ۱۵۳ را برگرداند. در نوشتن این تابع از بردار یا رشته استفاده نکنید، اما می‌توانید از پارامترهای اضافه استفاده کنید.

۳. پردازش بازگشتی لیست‌ها

در تمام این تمرین‌ها در صورت نیاز می‌توانید برای تابع خواسته شده پارامترهای دیگری هم تعریف کنید.

تمرین ۷ – تابعی بازگشتی بنویسید که یک بردار از اعداد صحیح را به عنوان ورودی بگیرد و تعداد اعداد مثبت آن را برگرداند.

تمرین ۸ – تابعی بازگشتی بنویسید که یک بردار از اعداد صحیح را به عنوان ورودی بگیرد و حاصل جمع اعداد مثبت آن را برگرداند. برای این کار تابع sum_list را که در بالا تعریف شده تغییر دهید.

تمرین ۹ – تابعی به نام to_upper بنویسید که یک رشته را به عنوان پارامتر بگیرد و همان رشته را برگرداند با این تفاوت که حروف کوچک به حروف بزرگ تبدیل شده است. سایر کاراکترها بدون تغییر باقی می‌ماند.

تمرین ۱۰ – یک رشته آینه‌ای است اگر خواندن آن از سمت چپ و راست یکسان باشد. مثلاً کلمه‌ی madam یک رشته آینه‌ای است. تابعی بازگشتی به نام palindrome بنویسید که یک رشته را بگیرد و در صورتی که آینه‌ای باشد مقدار true وگرنه false برگرداند. در این مرحله فرض کنید این رشته فقط از حروف الفبا تشکیل شده است.

تمرین ۱۱ – در تمرین قبل فرض کنید در رشته‌ی مورد نظر، کاراکترهای غیرحرفی هم وجود دارند که نباید در محاسبه‌ی آینه‌ای بودن در نظر گرفته شوند. همچنین، حروف بزرگ و کوچک هم یکسان در نظر گرفته می‌شوند. با این تعریف رشته‌ی "Madam, I'm Adam" آینه‌ای محسوب می‌شود. تابع palindrome2 را به شکل بازگشتی بنویسید که با تعریف جدید آینه‌ای بودن را تعیین کند.

تمرین ۱۲ – تابعی بازگشتی بنویسید که برداری به نام a را بگیرد و حاصل عبارت a[0] - a[1] + a[2] - ... ± a[n-1] را برگرداند (علامت جمله آخر براساس زوج یا فرد بودن n تعیین می‌شود).

تمرین ۱۳ – تابعی بازگشتی بنویسید که برداری به نام a را بگیرد و حاصل عبارت

a[0]*a[1] + a[2]*a[3] - ... + a[n-2]*a[n-1]

را برگرداند (در صورتی که n زوج باشد جمله‌ی آخر فقط a[n-1] خواهد بود).

تمرین ۱۴ – تابعی بازگشتی بنویسید که برداری به نام a را بگیرد و حاصل عبارت

a[0]*a[1] + a[2]/a[3] - ...

را برگرداند (در صورتی که n زوج باشد جمله‌ی آخر فقط a[n-1] خواهد بود).