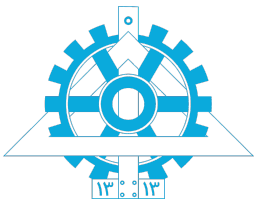
برنامه‌سازی پیشرفته | تمرین کامپیوتری دوم

زمان تحویل: ۱۴۰۴/۰۲/۰۱



ایمیل دستیاران آموزشی مربوطه:

زانکو کریمی، امیرمرتضی یوسفی، زانیار غزالی، محمدمهدی نیک‌خواه

مقدمه:

این تمرین با هدف آشنایی با مفاهیم برنامه‌نویسی بازگشتی (Recursion) طراحی شده و شامل سه تمرین مستقل است. توصیه می‌شود برای درک بهتر این مفاهیم، زمان کافی برای حل هرکدام در نظر گرفته شود. توجه داشته باشید که تمام تمرینات باید به روش بازگشتی حل شوند؛ حتی اگر راه‌حل‌های غیر بازگشتی نیز برای آن‌ها وجود داشته باشد. همچنین در دو سوال آخر، تنها بخشِ حل کلی مسئله و فرآیند بک‌ترکینگ (Backtracking) باید به‌صورت بازگشتی پیاده‌سازی شود و استفاده از حلقه‌ها در سایر بخش‌های آن سوالات مانعی ندارد.

تمرین‌ها:

تمرین اول:

برنامه‌ای بنویسید که با گرفتن دو عدد k و n (که k عددی صحیح و n عددی طبیعی است) و آرایه‌ای از اعداد صحیح به طول n، تمام جمع و تفریق‌هایی را که با اعداد موجود در آرایه ساخته و برابر با عدد k می‌شوند، پیدا کند.

مثلا اگر k = 2 و آرایه [1, 2, 3] باشد، جایگشت زیر یکی از حالات مطلوب است:

```
+ 3 − 2 + 1 = 2
```

ورودی:

در خط اول ورودی عدد n، سپس در خط بعدی آرایه‌ای از اعداد به طول n که اعداد آن با یک فاصله (space) از هم جدا شده‌اند، و در خط آخر عدد k داده می‌شود.

خروجی:

خروجی شما باید تعداد حالاتی که با جایگشت جمع و تفریق اعداد آرایه به عدد k می‌رسیم را نمایش دهد.

ورودی نمونه:

```
3
3 2 1
0
```

خروجی نمونه:

```
2
```

در نمونه بالا دو حالت خروجی به صورت زیر است: (این بخش در خروجی برنامه نشان داده نمی‌شود)

```
- 1 - 2 + 3
+ 1 + 2 - 3
```

تمرین دوم:

فرض کنید در سال ۱۹۸۵ هستید و به‌عنوان یکی از اعضای تیم توسعه‌ی اولیه‌ی ویندوز مایکروسافت، بیل گیتس بخش «جستجوی فایل‌ها در سیستم‌عامل» را به شما واگذار کرده است. وظیفه‌ی شما طراحی برنامه‌ای است که بتواند در ساختار فایل‌ها و فولدرها، یک فایل مشخص را با استفاده از روش **بازگشتی** پیدا کند و مسیر کامل آن، از **فولدر ریشه** (Root Folder) تا محل قرارگیری فایل را نمایش دهد.

ابتدا عدد صحیح n را دریافت می‌کنید که نشان‌دهنده‌ی تعداد کل فولدرها در سیستم است. سپس خطوط بعدی شامل اطلاعات مربوط به هر یک از فولدرها خواهد بود. اطلاعات ورودی هر فولدر به‌صورت زیر است:

```
<folder_name/>
<number_of_folders_and_files_inside_folder>
<folder_and_file_names>
...
```

به‌عنوان آخرین ورودی، نام فایلی که در جستجوی آن هستیم وارد می‌شود. خروجی نیز مسیری است که از ریشه به فایل مدنظر وجود دارد.

ورودی:

اسامی فایل‌ها و فولدرها به‌صورت رشته‌های بدون فاصله و یکتا هستند. توجه داشته باشید که هدف، جستجوی فایل‌ها است و به دنبال فولدرها نیستیم. در انتهای اسم هر فولدر «/» وجود دارد. فولدر اول ورودی همیشه فولدر ریشه (root/) است.

```
<num_of_folders>
<root/>
<num_of_folders_and_files_inside_folder>
<folder_and_file_names>
...
<folder_name/>
<num_of_folders_and_files_inside_folder>
<folder_and_file_names>
...
<file_name_to_be_searched>
```

ورودی نمونه:

```
4
root/
3
f1/
```

```
f2/
f3/
f1/
4
a.txt
b.txt
c.txt
d.txt
f2/
0
f3/
3
1.txt
2.txt
3.txt
2.txt
```

خروجی نمونه:

```
root/f3/2.txt
```

تمرین سوم:

فرض کنید یک ربات کاوشگر در یک پایگاه تحقیقاتی در قطب شمال وظیفه دارد با مقدار مشخصی باتری، از روی نقشه‌ای که به شکل جدولی مربعی است، از **نقطه‌ی شروع** حرکت کرده و در نهایت به **نقطه‌ی پایان** برسد. این ربات در مسیر خود باید حتما از تمام **ایستگاه‌های تحقیقاتی** حداقل یک بار بازدید کند. هدف شما طراحی برنامه‌ای است که تعداد تمام مسیرهای ممکن که این شرایط را رعایت می‌کنند، محاسبه کند.

نقشه به صورت یک ماتریس مربعی از رشته‌ها و اعداد (جداشده با فاصله) داده می‌شود. هر درایه می‌تواند یکی از موارد زیر باشد:

درایه	توضیحات	مصرف باتری
S	نقطه شروع (Start)	۳ واحد
F	نقطه پایان (Finish)	۱ واحد
R	ایستگاه تحقیقاتی (Research Station)	۲ واحد
a positive number	یک عدد مثبت	به‌اندازه عدد درایه

هنگامی که برنامه شروع می‌شود، روی خانه‌ی شروع هستیم و وقتی به یک خانه‌ی مجاور حرکت می‌کنیم، مقدار مصرف باتری مشخص‌شده‌ی آن خانه از باتری ربات کم می‌شود. (یعنی اگر از خانه S به خانه‌ی مجاور 2 برویم، دو واحد از باتری ربات کم می‌شود و اگر دوباره به خانه‌ی S برگردیم، سه واحد کاهش خواهیم داشت.) حرکت به صورت قطری ممکن نیست.

و در طی مسیر می‌شود از یک خانه دو بار رد شد. (به جز خانه پایانی)

مسیر زمانی معتبر است که از خانه‌ی S با حداقل یک بار عبور از خانه‌های R، بدون اینکه مقدار باتری ربات کمتر از صفر شود، به خانه‌ی F برسیم. (اگر پس از رسیدن به خانه‌ی F باتری ربات صفر شود، مسیر معتبر است.)

ورودی:

در خط اول مقدار اولیه‌ی باتری ربات و در خط‌های بعدی نقشه‌ی حرکت ربات به‌صورت ماتریس مربعی n×n داده می‌شود.

خروجی:

تعداد مسیرهای معتبر باید در خروجی نمایش داده شود.

ورودی نمونه:

```
6
S 1 R
2 1 1
1 F 3
```

خروجی نمونه:

```
2
```

نکات پایانی و نحوه تحویل:

- فایل تحویلی شما در ایلرن باید شامل سه فایل سی‌پلاس‌پلاس (.cpp) مستقل باشد که هر کدام مربوط به یکی از تمرین‌ها هستند. این سه فایل را در یک فایل زیپ (.zip) قرار دهید. فایل‌ها را با قالب روبه‌رو نام‌گذاری کنید:

```
<FirstName>_<LastName>_<StudentNumber>_CA2.zip
├── <FirstName>_<LastName>_<StudentNumber>_CA2_Q1.cpp
├── <FirstName>_<LastName>_<StudentNumber>_CA2_Q2.cpp
└── <FirstName>_<LastName>_<StudentNumber>_CA2_Q3.cpp
```

به طور مثال:

```
Amirreza_Akbari_810899000_CA2.zip
├── Amirreza_Akbari_810899000_CA2_Q1.cpp
├── Amirreza_Akbari_810899000_CA2_Q2.cpp
└── Amirreza_Akbari_810899000_CA2_Q3.cpp
```

دقت داشته باشید که آزمون‌های خودکار، فایل‌های شما را بر اساس این قالب نام‌گذاری اجرا می‌کنند. هرگونه مغایرت بین شیوه نام‌گذاری شما و شیوه ذکرشده در بالا، به معنی تصحیح‌نشدن فایل‌های شما و ازدست‌دادن نمره خواهد بود.

- برنامه‌ی شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++20 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- رعایت اصول کدنویسی تمیز و منظم در طراحی برنامه اهمیت زیادی دارد. نام‌گذاری معنادار و منسجم (Consistent) برای متغیرها و توابع، تقسیم کد به بخش‌های منطقی، نوشتن توابع کوتاه، عدم وجود کد تکراری و رعایت دندانه‌گذاری (Indentation) باعث افزایش خوانایی کد شما و اطمینان از صحت عملکرد برنامه می‌شود.
- وبسایت‌هایی مانند stackoverflow و cplusplus می‌توانند در حل چالش‌های این تمرین به یاری شما بیایند.
- درستی برنامه‌ی شما از طریق آزمون‌های خودکار سنجیده می‌شود. پیشنهاد می‌کنیم با استفاده از ابزارهایی مانند diff خروجی برنامه‌ی خودتان را با خروجی‌هایی که به‌عنوان نمونه در اختیارتان قرار گرفته مطابقت دهید.
- در پایان توجه داشته باشید که نمره‌ی این تمرین تنها به اجرای صحیح برنامه و مطابقت خروجی‌های شما با آزمون‌های خودکار وابسته نیست؛ بلکه ارائه حضوری و میزان تسلط شما در کد تحویلی‌تان نیز در نمره‌ی نهایی تاثیر خواهد داشت.