بسمه تعالی

دانشگاه فردوسی مشهد

تمرین Natural Merge Sort:

در این تمرین از دانشجو خواسته شده است که قسمتی از بدنه‌ی توابع Sort، FindBorder و Merge را در فایل NaturalMergeSort.h تکمیل کند:

```
Public void Sort(T[] A, int n){
	Queue<Integer> points = new LinkedList();
	findBorders(arr, n, points);
	//Write your code here
}
```

این تابع یک آرایه را به عنوان ورودی دریافت می‌کند و آن را با روش Natural Merge Sort و با کمک توابع FindBorder و Merge به صورت صعودی مرتب می‌کند. پارامتر های این تابع به این شرح است:

A: آرایه داده های ورودی　　n: تعداد داده های ورودی

```
Public void findBorder(T[] arr, int n, Queue<Integer> points){
	//Write your code here
}
```

این تابع یک آرایه را به عنوان ورودی دریافت می‌کند و اندیس نقاطی که در آن ترتیب صعودی رعایت نشده است و نقاط مورد نیاز دیگر برای انجام عمل ادغام را به عنوان خروجی باز می‌گرداند. پارامتر های این تابع به این شرح است:

arr: آرایه داده های ورودی　　n: تعداد داده های ورودی

points: صفی متشکل از اندیس نقاطی از آرایه که در آن ترتیب صعودی رعایت نشده و نقاط مورد نیاز دیگر برای انجام عمل ادغام است.

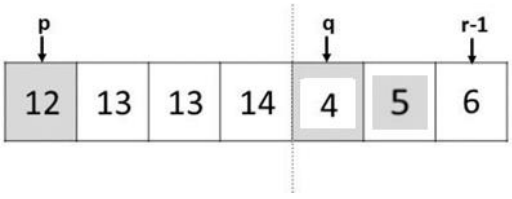
```
Public void merge(T[] arr, int p, int q, int r){
	//Write your code here
}
```

در این تابع عمل ادغام انجام می‌شود. برای انجام این عمل از آرایه‌ی temp به عنوان آرایه‌ی کمکی استفاده می‌شود. پارامتر های این تابع به شرح زیر است:

arr: آرایه‌ای که قرار است عمل ادغام روی آن انجام شود.

عناصر آرایه arr از p تا q-1 مرتب هستند.

عناصر آرایه arr از q تا r-1 مرتب هستند.



توضیحی در مورد ساختمان داده‌ی صف:

ساختمان داده‌ی صف ساختمان داده‌ای است که در آن عمل اضافه شدن عناصر به انتهای لیست و عمل حذف از ابتدای لیست انجام می‌شود. بهترین مثال برای آن همان صف در دنیای واقعی است که در آن اولین نفری که وارد صف شده، اولین کسی است که از صف خارج می‌شود. در این تمرین قرار است از این ساختمان داده که در ادامه‌ی درس با شیوه‌ی پیاده‌سازی آن به صورت دقیق‌تر آشنا می‌شویم، استفاده کنیم.

Push: با استفاده از این تابع می‌توان یک عنصر جدید را به انتهای صف اضافه کرد.

Pop: با استفاده از این تابع می‌توان عنصر سر صف را از صف خارج کرد.

Front: با استفاده از این تابع می‌توان به عنصر سر صف بدون خارج کردن آن از صف، دسترسی داشت.

نمونه برنامه ای که در آن از ساختمان داده‌ی صف استفاده شده:

```cpp
#include <iostream>
#include <queue>

using namespace std;

int main() {
	queue<int> q = queue<int>();

	cout << "push numbers 1 to 5." << endl;
	for (int i = 1; i < 6; i++)
		q.push(i);
	//--------------------------------
	cout << "pop 3 times: ";
	for (int i = 0; i < 3; i++) {
		cout << q.front() << " ";
		q.pop();
	}
	cout << endl;
	//--------------------------------
	cout << "push number 6" << endl;
	q.push(6);
	//--------------------------------
	cout << "pop other elements: ";
	while (q.size() > 0) {
		cout << q.front() << " ";
		q->pop();
	}
	cout << endl;
	return 0;
}
```

```
OUTPUT:
push numbers 1 to 5
pop 3 times: 1 2 3
push number 6
pop other elements: 4 5 6
```

در این تمرین نحوه ارزیابی به شرح زیر است:

۱. TestFindBorder: همانطور که از اسم این تست بر می آید هدف از این تست ارزیابی مرز (border) های بدست آمده است. از آنجایی که پیدا کردن درست این مرز ها مهم ترین نکته درNatural merge sort است، این تست ٪۱۰ از کل نمره را در بر دارد و همچنین به علت اهمیت فراوان آن، این تست پیش نیاز دیگر تست ها نیز است. بدین معنی که در صورت پاس نشدن این تمرین نمره ی دیگر تست ها نیز برابر با صفر خواهد شد.
۲. TestMerge: در این تست، تابع merge که عمل ادغام را انجام می‌دهد، بررسی می‌شود. این تست شامل ٪۱۵ نمره است.
۳. TestSortingRandomNumbers: در این تست بررسی می‌شود که آیا عملیات مرتب سازی (sort) به درستی انجام شده است یا خیر. در این تست چندین بار آرایه هایی با طول متفاوت به صورت تصادفی (random) مقداردهی می‌شوند و سپس توسط کد نوشته شده توسط شما مرتب سازی می‌شوند. این تست به تنهایی ٪۲۰ از کل نمره این تمرین را شامل می‌شود.
۴. TestOrder: این تست که ٪۲۵ از کل نمره را به خود اختصاص داده است همانند تست قبل درستی ترتیب آرایه مورد نظر را بررسی می‌کند اما با این تفاوت که تعداد آرایه هایی که باید مرتب سازی شوند و عمل مقایسه ای که بر روی آنها صورت می‌گیرد متفاوت است. هدف از این تست، بررسی مرتبه زمان اجرای کد شماست.
۵. TestTemplate: این تست بررسی می‌کند که آیا کد نوشته شده توسط شما می‌تواند داده هایی با ساختمان داده ای (data type) به غیر از اعداد (int) را نیز مرتب سازی کند یا خیر. این تست شامل ٪۱۰ از نمره ی کل است.
۶. TestComparisonCount: تعداد مقایسه های انجام شده در طول کد شما، در این تست سنجیده می‌شود. این تست ٪۲۰ از نمره کل را در بر دارد.

* پیشنیاز قبولی در همه تست ها، قبولی در دو تست TestFindBorder و TestMerge است.

* پیشنیاز قبولی در تست TestTemplate، قبولی در تست TestSortingRandomNumbers است.

* پیشنیاز قبولی در تست TestOrder ، قبولی در تست TestSortingRandomNumbers است.

* پیشنیاز قبولی در تست TestComparisonCount، قبولی در دو تست TestTemplate و TestSortingRandomNumbrs است.

* تمام کتابخانه‌های مورد نیاز برای حل مسأله در اختیار شما قرار گرفته است. افزودن کتابخانه جدید موجب رخ دادن compile Error در مرحله تصحیح کد می‌شود. بنابراین مجاز به استفاده از کتابخانه‌های متفرقه نیستید.

* در طول کد خود تنها یکبار می‌توانید تابع findBorder را فراخوانی کنید. بیش از یکبار فراخوانی این تابع، موجب از دست دادن نمره در تست TestFindBorder خواهد شد.

برای بارگذاری این تمرین گام‌های زیر را دنبال کنید :

۱- ابتدا فایل info.txt را با مشخصات خود پر کنید.

۲- پس از حل تمرین، از پوشه src همه فایل های اضافی که به دلیل کامپایل برنامه بوجود آمده اند را پاک نمایید. (ممکن است IDE شما به طور خودکار، فایل‌هایی را اضافه کند). در نهایت فقط فایل‌هایی که از ابتدا در پوشه src وجود داشته‌اند، همچنان باقی می‌مانند.

۳- پوشه src و فایل info.txt را در کنار این پوشه، زیپ کنید. مطمئن شوید که وقتی فایل zip را باز می کنید پوشه src و همچنین فایل info.txt را می بینید.

٤- دقت کنید که پسوند فایل شما حتما zip باشد و حجم فایل بالای یک مگابایت نباشد.

٥- فایل را در سامانه بارگذاری کنید.

٦- اشکالاتی را که سامانه مشخص کرده است برطرف نمایید و مجددا تمرین را در سامانه بارگذاری کنید.

۷- مرحله قبل را آن قدر ادامه دهید که از صحت عملکرد برنامه خود اطمینان حاصل نمایید.

با آرزوی موفقیت