بسمه تعالی

درس: ساختمان های داده ها دانشگاه فردوسی مشهد
مدرس: غیاثی شیرازی

تمرین Natural Merge Sort:

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

```
virtual void Sort (T* arr, int n){
        //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. پارامترهای این تابع به این شرح است:

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

```
void FindBorders(T* arr, int n, queue<int> &points){
        //Write your code here
}
```

این تابع یک آرایه را به عنوان ورودی دریافت می‌کند و اندیس نقاطی که در آن‌ها ترتیب صعودی رعایت نشده را در یک صف به عنوان خروجی باز می‌گرداند. پارامترهای این تابع به این شرح است:

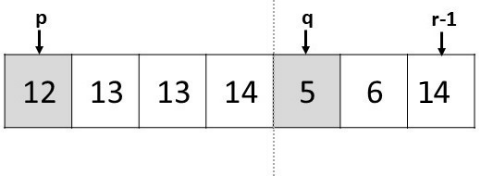
arr: آرایه داده های ورودی n: تعداد داده های ورودی

points: صفی شامل اندیس نقاطی از آرایه که در آن ترتیب صعودی رعایت نشده است.

```
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. پارامترهای این تابع به این شرح 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 ساختمان‌داده‌ی صف:

ساختمان‌داده‌ی صف ساختمان‌داده‌ای 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 کسی که وارد صف شده، اولین کسی است که از صف خارج می‌شود. در این تمرین قرار است از این ساختمان‌داده که در ادامه‌ی درس با شیوه‌ی پیاده‌سازی آن به صورت دقیق‌تر آشنا می‌شوید 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 دسترسی داشت.

size: با استفاده از این تابع می‌توان به اندازه‌ی فعلی صف دسترسی پیدا کرد.

نمونه برنامه‌ای که در آن از ساختمان‌داده‌ی صف استفاده شده است:

```
#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:

1. **TestFindborders**: 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 در Natural Merge Sort است این تست به تنهایی 40% 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.
2. **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 های متفاوت به صورت تصادفی (random) مقداردهی میشوند و سپس توسط کد نوشته شده توسط شما مرتب سازی میشوند. این تست به تنهایی 20%از کل نمره این تمرین را شامل میشود.
3. **TestOrder**: این تست که 30% 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.
4. **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 , double) را نیز مرتب سازی کند یا خیر. این تست شامل 10%از نمره ی کل است.

برای انجام این تمرین کارهای زیر را انجام دهید:

1- ابتدا در این پوشه فایل info.txt را با مشخصات خود پر کنید.
2- سپس به پوشه src بروید و پروژه را در Visual Studio باز کنید.
3- کد برنامه Natural Merge Sort را در فایل NaturalMergeSort.h تکمیل کنید.
4- برنامه را بر روی تست(های) داده شده تست نمایید.
5- در صورت تمایل تست های بیشتری بنویسید تا از عملکرد برنامه خود اطمینان بیشتری حاصل نمایید.
6- برنامه را اجرا کنید و پس از اطمینان از صحت عملکرد آن، با استفاده از کلید PrtScr از خروجی برنامه عکس بگیرید.
7- عکس را با استفاده از mspaint در پوشه img ذخیره نمایید.
8- از پوشه src همه فایل های اضافی که به دلیل کامپایل برنامه بوجود آمده اند را پاک نمایید. (پوشه Debug و فایل با پسوند sdf را حتما پاک کنید زیرا حجم زیادی می گیرند).
9- محتویات کل پوشه را به صورت یک فایل zip در آورید.
10- مطمئن شوید که وقتی فایل zip را باز می کنید پوشه های src, img و test و همچنین فایل info.txt را می بینید.
11- نام این فایل zip را به «شماره تمرین-شماره دانشجویی» تغییر دهید.

12- ابتدا این فایل را به سیستم «سپهر» ایمیل کنید تا از نحوه عملکر برنامه خود بر روی تست های تکمیلی آگاه شوید.
13- اشکالاتی را که سیستم «سپهر» مشخص کرده است برطرف نمایید و مجددا تمرین را به سیستم «سپهر» تحویل دهید.
14- مرحله قبل را آن قدر ادامه دهید که از صحت عملکرد برنامه خود اطمینان حاصل نمایید.
15- نسخه نهایی فایل zip خود را تهیه نمایید.
16- این فایل را از طریق سیستم vu تحویل دهید.

با آرزوی موفقیت