دانشگاه صنعتي اصفهان

دانشكده مهندسي برق و كامپيوتر

تکلیف دوم درس ساختمان داده‌ها

استاد : دکتر عبدالرضا میرزایی

مسئول تمرین : پویا بهزادی فر

سوال یک - کلاس پشته بر پایه لیست پیوندی

هدف کلی این سوال پیاده سازی کلاس پشته با استفاده از ساختمان داده لیست پیوندیست که بالاترین عنصر پشته را به انتهای لیست پیوندی اضافه میکند.

کدی که شما باید بنویسید

شما باید پیاده سازی کلاس پشته به وسیله ی ساختمان داده‌ی مشخص شده را کامل کنید.
ساختمان داده‌ی مشخص شده یک لیست پیوندی یک طرفه است که داده push شده در انتهای (tail) آن اضافه میشود.(به طور مشابه عناصر از انتهای لیست پیوندی pop میشوند.)

به دانشجویانی که در پیاده سازی آن ها داده جدید در ابتدای لیست پیوندی (head) اضافه شود نمره ای تعلق نخواهد گرفت.
پیاده سازی خود را در فایل stack.cpp ذخیره کنید.

سوال دو - بررسی بهینگی زمانی

با توجه به پیاده سازی خود در سوال یک بررسی کنید push کردن n داده جدید در پشته چه پیچیدگی زمانی دارد.(پیچیدگی را به وسیله Big O notation بیان کنید.)

سپس بررسی کنید pop کردن n داده ای که در مرحله قبل push کردید چه پیچیدگی زمانی دارد و آن را به وسیله ی Big O notation بیان کنید.
در این سوال پاسخ نهایی کافی نمی باشد و باید پاسخ خود را مرحله به مرحله بیان کنید.

پاسخ خود را در یک pdf با نام Analysis.pdf ذخیره کنید.

سوال سه - محاسبه عبارت infix

اگرچه میتوان از پشته برای محاسبه‌ی عبارت‌های پسوندی(postfix) استفاده کرد، اما واقعیت این است که اکثر افراد عبارات را به این شکل نمی نویسند و از عبارات infix استفاده میکنند.

الگوریتمی وجود دارد که میتوان به وسیله ی آن عبارت infix را محاسبه کرد که به دو پشته نیاز دارد: یکی برای اعداد و دیگری برای عملگرها.

تصمیم‌ها بر اساس توکن ورودی بعدی(T) که میتواند یک عملگر، عدد و یا EOF باشد و عنصر top پشته عملگرها گرفته خواهد شد.

```
while T is not EOF or the operator stack is non empty

    if T is a number:
        push T to the number stack; get the next token

    else if T is a left parenthesis:
        push T to the operator stack; get the next token


    else if T is a right parenthesis:
        if the top of the operator stack is a left parenthesis:
            pop it from the operator stack; get the next token:
        else:
            pop the top two numbers and the top operator
            perform the operation
            push the result to the number stack

    else if T is +, - or EOF:
        if the operator stack is nonempty and the top is one
                                            of +, -, *, /:
            pop the top two numbers and the top operator
            perform the operation
            push the result to the number stack
        else:
            push T to the operator stack; get the next token

    else if T is * or /:
        if the operator stack is nonempty and the top is one of *, /:
            pop the top two numbers and the top operator
            perform the operation
            push the result to the number stack
```

```
else:
    push T to the operator stack; get the next token
```

وظیفه شما در این سوال پیاده سازی این الگوریتم با زبان cpp است.

الزامات

- برنامه‌ی شما یک عبارت محاسباتی استاندارد را از ورودی خوانده و حاصل آن را در خروجی نشان خواهد داد.
- شما باید الگوریتم بالا را به وسیله‌ی پشته پیاده سازی کنید در نتیجه استفاده از systemcall و یا parser هایی که توسط شما پیاده سازی شده‌اند مجاز نمیباشد و در صورت استفاده نمره صفر به کد شما تعلق خواهد گرفت.
- تمامی اعداد ورودی صحیح و مثبت میباشند.
- فایل خود را با نام Eval.cpp ذخیره کنید.

راهنمایی

- یک کلاس Scanner در اختیار شما قرار گرفته که توکن هارا برای شما تولید میکند.
  لطفا فایل Scanner.h را مطالعه کرده تا متوجه عملکرد آن شوید.
- شما از کلاس پشته‌ای که در اختیارتان قرار داده شده به عنوان پشته اعداد و عملگرها استفاده خواهید کرد.
- چند نمونه عبارت برای تست کردن کدتان قرار داده شده است. شما نیز باید چند عبارت برای تست کردن کدتان قرار دهید.

سوال چهار - گسترش یک صف (Circular Array)

در فایل‌هایی که در اختیار شما قرار گرفته یک کلاس Queue که به وسیله آرایه پیاده سازی شده قرار گرفته است. اگر فایل main را build و اجرا کنید متوجه خواهید شد صف به اندازه کافی بزرگ نشده و تعدادی از عناصر از دست خواهند رفت.
شما میتواند مقدار پارامتر INITIAL_CAPACITY را افزایش دهید ولی این کار در همه ی شرایط جواب نخواهد داد. برای مثال شما نیاز دارید گنجایش صف به مقدار مناسب زیاد شود که به یک استراتژی نیاز دارد. استراتژی مورد نظر صف را به اندازه‌ای که ما به آن نیاز داریم به صورت پویا افزایش خواهد داد.

۱.فایل Queue.h را باز کرده و data type آرایه elements را به پوینتر تغییر دهید.
۲.سپس یکی از methodهای Queue.cpp (کانستراکتور؟ یا enqueue؟ یا ...) را به نحوی تغییر دهید که آرایه را به صورت پویا ایجاد کند.
۳.کلاس هایی که از حافظه گیری پویا استفاده میکنند به destructor مناسب نیاز دارند. آن را پیاده سازی کنید.
۴.تغییر سایز آرایه عملیات پر هزینه‌ای است. پیدا کردن فضای جدید و کپی کردن عناصر از آرایه قبلی به آرایه جدید و پاک کردن آرایه قبلی که این کار از اردر n میباشد.
یک استراتژی مناسب دو برابر کردن ظرفیت آرایه زمانی است که حافظه آرایه پر میشود. این استراتژی را با دوباره نوشتن متد enqueue پیاده کنید.
۵.داشتن یک آرایه که ظرفیت آن در مقایسه با تعداد عناصر ذخیره شده بسیار بیشتر باشد هم نامناسب است زیرا باعث هدررفت حافظه خواهد شد. یک استراتژی مناسب نصف کردن ظرفیت آرایه زمانی است که عناصر ذخیره شده کمتر از یک چهارم ظرفیت آرایه را مصرف کرده‌اند. با این وجود ظرفیت صف نباید کمتر از مقدار INITIAL_CAPACITY شود. این استراتژی را با دوباره نوشتن تابع dequeue پیاده کنید.

نحوه نمره دهی

در تمامی سوالات علاوه درستی پاسخ، نحوه پاسخ دهی و پیاده سازی نیز حائز اهمیت است. برای مثال بهینگی، coding style مناسب و کامنت گذاری برای توابع پیاده سازی شده باید رعایت شود.

در مورد گروه‌ها

شما میتوانید تکلیف را در گروه‌های دونفره انجام دهید.
انجام تکلیف به صورت فردی مانعی ندارد.
پاسخ خود را با فرمت HW2_STU1NUM_STU2NUM_STU1NAME_STU2NAME اپلود کنید.
اپلود کردن پاسخ توسط یکی از اعضای گروه کافیست.