به نام خداوند بخشنده ی مهربان

ساختمان های داده

دکتر سید ابوالقاسم میرروشندل

طراح : آرش یوسفی

پروژه ی دوم

داستان پروژه:

" پدرو خوزه (Pedro José)" از اهالی اسپانیاست. گفته می شود که او در جوانی، در حوزه ریاضیات برای خودش صاحب سبک بوده و حتی بُعد چهارم را می دیده است! ولی چند سالی است که به علت مشکلات زندگی ریاضیات را کنار گذاشته و در اطراف کوه Teide به چوپانی مشغول است. دیروز وقتی که پدرو در زیر سایه درختی مشغول وب گردی بود، ناگهان با یک نرم افزار کاربردی مواجه شد که تعدادی از محاسبات ریاضی را انجام می داد! در همین لحظه، پدرو دچار دگرگونی درونی شده و تصمیم میگیرد مشابهِ داخلیِ این محصولِ خارجی (!!) را به دستان توانمندِ داخلیِ خودش تولید کند! ولی پدرو ریاضیات را تا حدودی فراموش کرده و برنامه نویسی هم بلد نیست! و حالا از شما می خواهد تا به او کمک کنید و یک محصول مشترک اسپانیا-ایران تولید نمایید!

هدف پروژه:

هدف از این پروژه، در گام نخست، آشنایی دانشجویان با ساختمان‌داده های لیست‌پیوندی، پشته، صف و پیاده‌سازی هر یک از آنهاست.

در قدم دوم، آشنایی با درخت عبارت و الگوریتم های مرتبط با آن مدّنظر است.

ساخت ساختمان های داده:

ابتدا باید داده ساختار (همان Data structure !) های زیررا مطابق با دستورالعمل داده شده(!!) پیاده سازی نمایید. **در پیاده سازی ها می توانید از نوع های داده ای عام (Generic ها) در تعریف کلاس استفاده کنید.**

۱-ساختمان داده لیست پیوندی (Linked List) را طبق مباحث درس پیاده سازی کنید.لیست پیوندی ساخته شده باید دارای متدهای زیر باشد:

- Void addFirst(Element e) : عنصر e را به ابتدای لیست پیوندی اضافه می کند.
- Void addLast(Element e) : عنصر e را به انتهای لیست پیوندی اضافه می کند.
- Void deleteFirst( ) : عنصر ابتدای لیست را حذف می کند(در صورتی که عنصری موجود باشد).
- Void deleteLast( ) : عنصرانتهای لیست را حذف می کند(در صورتی که عضوی وجود داشته باشد).
- Element getFirst( ) : عنصر ابتدای لیست را برمیگرداند(در صورتی که لیست خالی است ، NULL برمیگرداند).
- Element getLast( ) : عنصر انتهای لیست را برمیگرداند(در صورتی که لیست خالی است ، NULL برمیگرداند).
- Long int getSize( ) : تعداد عناصر داخل لیست را برمیگرداند.
- Void clear( ) : لیست را خالی می کند.
- Boolean isEmpty() : در صورت خالی بودن لیست true وگرنه false برمیگرداند.

تمامی عملیات های لیست پیوندی باید با پیچیدگی زمانی O(۱) انجام شود.

۲-با استفاده از لیست پیوندی ، ساختمان داده صف (Queue) را با توابع زیر ایجاد کنید:

- Void enqueue(Element e) : عنصر e را به ابتدای صف اضافه می کند.
- Element dequeue( ) : عنصر ابتدای صف را حذف می کند وآن را برمیگرداند(در صورتی که عنصری موجود باشد).
- Element getFirst( ) : عنصر ابتدای صف را برمیگرداند(در صورتی که صف خالی است ، NULL برمیگرداند).
- Long int getSize( ) : تعداد اعضای داخل صف را برمیگرداند.
- Void clear( ) : صف را خالی می کند.
- Boolean isEmpty() : در صورت خالی بودن صف true وگرنه false برمیگرداند.

تمامی عملیات های صف باید با پیچیدگی زمانی O(۱) انجام شود.

۳-برای ساخت پشته(Stack) می توانید از صف یا لیست پیوندی استفاده کنید. پشته باید از توابع زیر پشتیبانی کند :

- Element pop( ) : عنصر بالای پشته را حذف می کند وآن را برمیگرداند(در صورتی که پشته خالی نباشد).
- Void push(Element e): عنصر e را در بالای پشته قرار می دهد.
- Element peek( ) : عنصر بالای پشته را برمیگرداند(در صورتی که پشته خالی است ، NULL برمیگرداند).
- Long int getSize( ) : تعداد اعضای داخل پشته را برمیگرداند.
- Void clear( ) : پشته را خالی می کند.
- Boolean isEmpty() : در صورت خالی بودن پشته true وگرنه false برمیگرداند.

تمامی عملیات های پشته باید با پیچیدگی زمانی O(۱) **انجام شود.**

بخش های اصلی برنامه :

برنامه از تعدادی دستور کنترلی تشکیل می شود. دستورات کنترلی با کاراکتر # شروع می شوند و هر آنچه که در ادامه ی دستور در همان خط می آید مربوط به آن دستور است.در صورت زدن کلید Enter برنامه دستور فعلی را پایان یافته تلقی می کند. برای ایجاد دستور جدید باید به خط بعد رفت.بعد از Run کردن برنامه می توانیم از دستورات کنترلی زیر استفاده کنیم :

۱-دستور #new محیط کار جدیدی را ایجاد می کند.تمامی تعاریفِ متغیر ها و محاسبات محیط های قبلی از بین می روند.دقت کنید که بعد از اجرای برنامه شما ، بدون نوشتن دستور #new کار کردن با برنامه آغاز نمی شود.

۲-دستور #define برای تعریف متغیرهای صحیح و اعشاری و توابع گویا به کار می رود.توابع دارای آرگومان می باشند.دقت کنید که در تعریف متغیر ها و توابع ، علاوه بر عملگرها و عملوندها ، پرانتز گذاری برای تعیین اولویت عملگرها نیز وجود دارد و شما باید با استفاده از درخت عبارت و الگوریتم های آن ، مقدار درست را به توابع و متغیرها نسبت دهید.نکته دیگر اینکه متغیرها حتما باید هنگام تعریف ، مقداردهی شوند.

عبارت های ریاضی که ما آن ها را می شناسیم معمولاً به صورت میانوندی نوشته می شوند. **عبارت هایی که در این برنامه داده می شوند از یک یا چند عملوند ، با تعدادی عملگر و پرانتز تشکیل شده اند. عملوند ها تشکیل شده از اعداد صحیح یا اعشاری، توابع به ازای مقداری خاص، و یا متغیر هایی هستند که مقادیر صحیح یا اعشاری به خود گرفته اند.** عملگرها می‌توانند Unary یا Binary باشند. عملگرهای Binary شامل **جمع(+)** و **تفریق(-)** و **ضرب(*)** و **تقسیم(/)** و **توان(^)** هستند و تنها عملگر Unary که در این برنامه استفاده می شود علامت **منفی(-)** است.

فرم کلی استفاده از این دستور به صورت زیر است :

#define *<variable name or function name> = < arithmetic expression>*

در مورد این دستور ، قوانین زیر باید رعایت شوند :

- در صورتی که عبارت ریاضی (هم مقداردهی متغیر و هم تعریف تابع) بی معنی باشد، باید پیغام خطای مناسب طبق فرمت زیر چاپ شود:

>>err1 : wrong exp

علاوه بر آن ، تمام این خط از برنامه باید نادیده گرفته شده و روند عادی برنامه طی شود.

- برای فاصله گذاری بعد از نوشتن کلمه define، تنها یک واحد فاصله کافی است اما تعداد بیشتری فاصله غلط محسوب نمی شود.
- دقت کنید که اگر در برنامه از متغیر یا تابعی استفاده می‌کنید، حتما باید قبلا آن را تعریف کرده باشید و چنانچه متغیر یا تابعی پیش از تعریف، مورداستفاده قرار گیرد باید کل دستورِ اجرایی، نادیده گرفته شود و پیغام خطای مناسب در خط بعدی با فرمت زیر چاپ گردد:

>>err2 : undefined exp

- در نامگذاری متغیرها و توابع تنها استفاده از حروف کوچک و بزرگ الفبای انگلیسی مجاز است.
- متغیر های آرگومان توابع فقط در داخل تابع اعتبار دارند و هم نامی آن ها با سایر متغیرها، توابع دیگر و یا متغیر های سایر توابع به معنی یکی بودن آن ها نیست.
- مقدار متغیرها و تعریف توابع در حین اجرای برنامه و بعد از تعریف اولیه آن ها قابل تغییر است؛ فرض کنید متغیری به نام x را برای اولین بار تعریف می کنید و به آن مقداری می دهید؛ به طور مثال :

#define x = 7

بنابراین از این پس، x متغیری با مقدار ۷ است.حالا در صورتی که در ادامه دستور زیر نوشته شود :

#define x = 3

مقدار جدید x برابر ۳ می شود.بنابراین تغییر مقدار متغیرها هم با کمک این دستور انجام می شود.این نکته برای توابع هم صادق است.

- ممکن است نحوه پرانتزگذاری و یا استفاده از عملگرها و عملوندها در دستورات از نوع define نادرست باشد.در این صورت برنامه باید به صورت زیر خطا را نمایش داده و این خط از کد را درنظر نگیرد و به اجرای برنامه ادامه دهد :

>>err3 : wrong syntax

- در این برنامه، توابع گویا می توانند مقادیر خارج از دامنه ی تعریفِ خود را نیز بپذیرند و مقدارِ "تعریف نشده" اختیار کنند؛ در این حالت، برای نمایش مقدارِ تعریف نشده از عبارت undefined value مطابق فرمت زیر استفاده می شود:

>>undefined value

مثال های دستور define :

#define x = 12

#define y = 13.77

#define f(x) = 3*x – 12/4*3

#define g(x) = 2^x – 1

#define g(x,t) = x*t

#define P(x,y,z) = x-y+z

#define arashYousefi(x,t) = 1/2*x^2 + 3*t-1

…

۳-دستور #print که به صورت زیر فراخوانی می شود:

#print *<arithmetic expression>* or *<variable name>*

این دستور مقدار نهایی عبارت (value) جلوی print را در خط بعدی با فرمت زیر چاپ می‌کند:

>>value

مانند قسمت های قبلی ، برای خروجی دادن باید به خط بعد رفته و بعد از نمایش علامت ">>" ، خروجی موردنظر را چاپ کرد.

عبارت جلوی دستور print یک عبارت ریاضی حاوی تعدادی محاسبه ریاضی است. انواع محاسبات ریاضی که در این نرم‌افزار می توان از آن ها استفاده کرد در دسته های زیر قرار می گیرند :

الف) عبارت های پرانتزگذاری شده شامل عملگر ها(که قبلا توضیح داده شدند) و عملوند ها.به طور مثال :

#print 12*3+2

>>۳۸

#print (2+3)*(3/2)

>>7.5

ب) عملیات های ریاضی که روی توابع گویا صورت می گیرد.شامل موارد زیر :

- انقباض یا انبساط و انتقال توابع.برای نمونه قطعه کد زیر را در نظر بگیرید:

```
#define f(x) = 9*x-1

#print 7-f(1)*1/2

#print  f(-1)/2-1
```

که خروجی آن به صورت زیر است:

```
>>3

>>-6
```

- ترکیب توابع.در این مورد به جای متغیر یا عدد در آرگومان، تابع با مقداری مشخص یا یک عبارت ریاضی پیچیده در آرگومان قرار می گیرد.ترکیب توابع می تواند به تعداد زیادی انجام شود.برای مثال فرض کنید دو تابع f(x) و g(x) داشته باشیم .برای ترکیب آن ها از نماد (g ∘ f)(x) استفاده نمی کنیم و برای این عملیات از نماد g(f(x)) و مشابه آن استفاده می شود. برای نمونه ی بیشتر مثال های ورودی و خروجی را ببینید.

- جمع، تفریق، ضرب و تقسیم توابع به ازای مقادیر خاصی که به آن ها پاس می شود.برای مطالعه بیشتر، اینجا را ببینید.

٤-دستور #solve variable برای حل معادله های درجه ۱ و۲ که تنها بر حسب یک متغیر هستند به کار می رود.متغیر معادله شامل یکی از حروف کوچک الفبای انگلیسی خواهد بود که در فرمت کلی به جای variable می نشیند. فرم کلی استفاده از دستور :

```
#solve  < variable>   <arithmetic equation>
```

این دستور بعد از حل معادله جواب یا جواب‌های آن را (در صورت وجود در مجموعه اعداد حقیقی) به فرم زیر چاپ می کند :

variable = answer **or** >>variable = answer1 , answer2 **or** >>variable = undefined in real numbers

مثال های این دستور:

```
#solve  x  x^2-4=0

>>x = 2, -2

#solve  x  7-3*x+1 = -1

>>x = 3
```

٤-دستور end# برای خروج از برنامه استفاده می شود.

فرمت ورودی و خروجی :

برنامه باید به صورت مداوم با کاربر در ارتباط بوده و به ازای دریافت دستورات کنترلی مختلف، پاسخ مناسب دهد.همچنین برای بعضی از دستورات که به خروجی دادن نیاز دارند،در محیط Console ، باید به خط بعد رفته و بعد از نمایش علامت "<<" ، خروجی موردنظر را چاپ کرد.

مثال های ورودی و خروجی :

INPUT	OUTPUT
#new #define x = 7 #define y = 1 #print y^x #print (2+3)*3/(2(#print x #end	>>1 >>err3 : wrong syntax >>7
#define u = 10 #print u+1 #print u*u	
#new #define f(x) = x + 2 #define t = 1 #print f(f(f(f(0)))) #end	>>8

#new #define g(t) = 2^t-1 #print g(2^2-1*2) #print g(2^1^3^3^12) #end	>>3 >>3
#new #define g(t) = 1 #define e(u) = u -1 #define A(t) = t #print A(g(e(1))) #print 2-3*1/1 #end	>>1 >>-1
#new #define r(t) = 1/(x^2-2*x) #print 1 #print r(2) #print 3-2*1 #end	>>1 >> undefined value >>1

جزئیات پیاده سازی:

حتماً باید ساختمان داده های لیست پیوندی و صف و پشته را با ویژگی های خواسته شده پیاده‌سازی کنید؛ حتی اگر در روند پروژه از آن ها استفاده نمی کنید! استفاده از آرایه برای پیاده سازی ها بدون مانع است.

دقت کنید که رعایت یک واحد فاصله میان بخش های ذکر شده در فرمت های کلی دستورات الزامی است؛ اما تعداد بیشتری فاصله غلط محسوب نمی شود.

نکات تکمیلی:

- پیاده سازی امکانات جدید و گسترش برنامه، در صورت صلاح دید دوستان حل تمرین، **نمره ی مثبت** خواهد داشت.
- برای مطالعه درباره اولویت عملگرها می توانید به این لینک مراجعه کنید.
- قطعه کدهای نمونه و خروجی آن ها در کنار این پروژه در اختیار شما قرار خواهند گرفت.
- پیاده سازی به صورت تک نفره است و هیچ محدودیتی برای زبان برنامه نویسی وجود ندارد.
- همه ی ساختمان داده های مورد نیاز، باید با توجه به تعریف پروژه پیاده سازی شوند. استفاده از ساختمان داده های آماده مجاز نیست.
- بحث و بررسی میان دانشجویان آزاد است اما هر دانشجو موظف است پروژه را به تنهایی انجام دهد و در هنگام تحویل حضوری، به تمام جزئیات کد کاملاً مسلط باشد. با موارد تقلب و کپی کردن، طبق تشخیص دوستان حل تمرین، برخورد جدی خواهد شد.
- توجه کنید که کدهای شما باید خوانا و دارای کامنت گذاری مناسب باشد.
- در کلاس کوئرای **Data Structures 971** با رمز **guilan96** ثبت نام کرده، نام و شماره ی دانشجویی خود را به درستی وارد کنید. عواقب بی دقتی در این مورد، به عهده ی دانشجو است.
- برای پرسش و پاسخ درباره ی پروژه، فقط از سامانه ی کوئرای درس مربوطه، اقدام کنید.
- پوشه ی مربوط به کد پروژه را (در صورت نیاز همراه با فایل pdf شرح انجام پروژه، نحوه ی اجرای برنامه، گزارش مربوط به تحلیل ساختمان داده ها و محاسبات انجام شده) در قالب یک فایل zip و در بخش مربوطه بارگذاری کنید. عواقب بی دقتی در این مورد، به عهده ی دانشجو است.
- زمان بندی و چگونگی تحویل حضوری پروژه، متعاقباً اعلام خواهد شد.