به نام خدا

|  |  |  |  |
| --- | --- | --- | --- |
| مدرس: مظفر بگ محمدي | دانشگاه ايلام | ترم اول سال تحصيلي 01-00 | سری سوم تمرینات ساختمان داده |

1. **پشته**: تعدادی عنصر در یک پشته قرار دارند. با استفاده از دو پشته‌ی دیگر عناصر پشته ی اول را مرتب کنید.
2. پشته و صف: با استفاده از دو پشته یک صف درست کنید. فقط لازم است که عملکردهای Enqueue و Dequeue را پیاده کنید.
3. پشته: فرض کنید رشته ی s حاوی یک جمله ی انگلیسی است. با استفاده از یک پشته ترتیب کلمات داخل جمله را معکوس کنید. راهنمایی: کاراکترهای s را تا وقتی که به whitespace برنخورده‌‌اید در داخل پشته بریزید، سپس پشته را در خروجی خالی کنید و ادامه دهید. مثال: data structure به atad erutcurts تبدیل می ‌شود.
4. پشته‌: با استفاده از یک پشته ی کمکی تمام عناصر پشته ی ورودی را به صورت مرتب به پشته ی خروجی منتقل کنید. در پایان، پشته کمکی باید خالی باشد.
5. پشته‌: یک رشته از کاربر دریافت کنید و بیشترین تعداد پرانتزهای معتبر این رشته را چاپ کنید.
   * مثلا اگر ورودی به صورت “((ds981((ilam university))))(((((((((data structure)))” باشد تعداد 8 پرانتز معتبر داریم!
6. پشته‌: یک رشته از پرانتزها از کاربر دریافت کنید و مینیمم تعداد پرانتزهای لازم برای معتبر شدن آن را نمایش دهید!
   * مثلا اگر ورودی به صورت “()))((” باشد تعداد 4 پرانتز دیگر لازم است تا این مجموعه معتبر شود.
7. پشته‌: طول بزرگترین زیردنباله نزولی را در یک آرایه پیدا کنید و زیر دنباله را نمایش دهید.
8. صف: در این تمرین شما باید با استفاده از یک پشته عناصر صف ورودی را در صف خروجی مرتب کنید.
9. صف: یک متد بنویسید که با استفاده از متدهای استاندارد صف جای عنصر اول و آخر صف را با هم عوض کند. فرض کنید اندازه ی صف بزرگتر یا مساوی دو است. می توانید از یک صف کمکی استفاده کنید.
10. صف: یک صف با طول زوج را در نظر بگیرید. عناصر نیمه ی اول صف را با عناصر نیمه دوم به صورت زیر قاطی کنید. شما فقط مجاز به استفاده از ساختار داده ی صف هستید.

Input : 1 2 3 4

Output : 1 3 2 4

Input : 11 12 13 14 15 16 17 18 19 20

Output : 11 16 12 17 13 18 14 19 15 20

1. پشته: با استفاده از دو پشته عناصر یک آرایه را به روش bubble مرتب کنید. می توانید از الگوریتم زیر استفاده کنید:

1. Push all elements of array in 1st stack

2. Run a loop for 'n' times(n is size of array) having the following :

2.a. Keep on pushing elements in the 2nd stack till the top of second stack is

smaller than element being pushed from 1st stack.

2.b. If the element being pushed is smaller than top of 2nd stack then swap them

(as in bubble sort)

\*Do above steps alternatively

TRICKY STEP: Once a stack is empty, then the top of the next stack will be the largest

number so keep it at its position in array i.e arr[len-1-i] and then pop it from that

stack.

1. پشته: پشته در ارزیابی عبارات ریاضی بسیار موثر است. معمولا عبارات ریاضی به فرم میانوندی infix داده می شوند و اولویت عملگرهای ریاضی به همراه پرانتز بندی نحوه ی محاسبه ی عبارت ریاضی را تعیین می کند. به عنوان مثال، عبارات ( A + B )\*C و A + ( B \* C ) دارای معانی متفاوتی هستند. در کامپیوتر به جای فرم میانوندی از فرم پسوندی postfix استفاده می شود. به عنوان مثال، معادل پسوندی عبارت (A+B) با AB+ برابر است. محاسبه ی عبارات پسوندی با استفاده از پشته راحت است. لذا، ابتدا باید عبارت میانوندی را به معادل پسوندی آن تبدیل کنیم. سپس، با استفاده از روش زیر عبارت را محاسبه کنیم. دقت کنید که عملگر ^ دارای بالاترین اولویت است. اولویت \* و / از + و – بیشتر است. در شرایط برابر، اولویت عملگرها از چپ به راست تعیین می شود. لذا، معادل پسوندی عبارت (A-B)\*[C/(D+E)+F] با AB- CDE +/F +\* برابر است. الگوریتم ارزیابی یک عبارت پسوندی به صورت زیر است:
2. Push the operands into the stack in the order they are appear.
3. When any operator encounter then pop two topmost operands for executing the operation.
4. After execution push the result obtained into the stack.
5. After the complete execution of expression, the final result remains on the top of the stack.

به عنوان مثال، عبارت میانوندی (2+4) \* (4+6) به عبارت پسوندی 2 4 + 4 6 + \* تبدیل و طبق شکل زیر محاسبه می شود:

