سوال ۳– ابتدا شبه کد لازم برای مرتب سازی آرایه arr که در زیر داده شده است را بنویسید. سپس پیچیدگی زمانی شبه کد خود را بدست آورید. (ارائه‌ی توضیحات الزامی است)

arr[] = 2, 1, 4, 3, 6, 5,....i, i-1, ...n, n-1

به صورت کلی می‌تونیم با درخت تصمیم اثبات کنیم که کران پایین الگوریتم‌های مرتب‌سازی مقایسه‌ای آرایه‌ای که ترتیب رندوم دارد، از $\Omega(nlogn)$ می‌باشد. ولی با اندکی دقت در آرایه داده شده متوجه می‌شویم که آرایه دارای یک الگو خاص می‌باشد.
آرایه داده شده نسبتاً مرتب شده است به جز اینکه جفت عناصر متوالی برعکس هستند. پس برای مرتب‌سازی آرایه صرفاً به جابه‌جا کردن جفت عناصر متوالی نیاز داریم.

```
Function sort_the_arr(arr):
     n = length(arr)

     for i from 0 to n-1 each step 2:
          if arr[i] > arr[i+1]:
               swap(arr[i], arr[i+1])

     return arr
```

پیچیدگی زمانی:
این الگوریتم یک بار طول آرایه را با گام‌های دوتایی پیمایش می‌کند و این حلقه به تعداد $n/2$ تکرار می‌شود که در هر تکرار کاری از زمان ثابت($O(1)$) انجام می‌دهد(شرط مقایسه و جابه‌جا کردن دو عنصر). در نتیجه این الگوریتم دارای پیچیدگی زمانی خطی و یا به عبارتی از $O(n)$ می‌باشد.
پیچیدگی حافظه:
همانطور که مشخص است این الگوریتم به صورت درجا انجام می‌شود و حافظه‌ای اضافه نیاز ندارد. در نتیجه از $O(1)$ است.

نکته: لازم به ذکر است این الگوریتم مشخصاً برای آرایه داده شده و فرض هایی از ساختار آرایه که بالاتر گفته شده درست عمل می‌کند. مثلاً اگر فرض می‌کردیم که اکثر آرایه‌هایی که داده می‌شود تقریباً به فرمت داده شده هستند، آنگاه شاید بهتر بود با تغییراتی از ایده Bubble sort استفاده می‌کردیم.