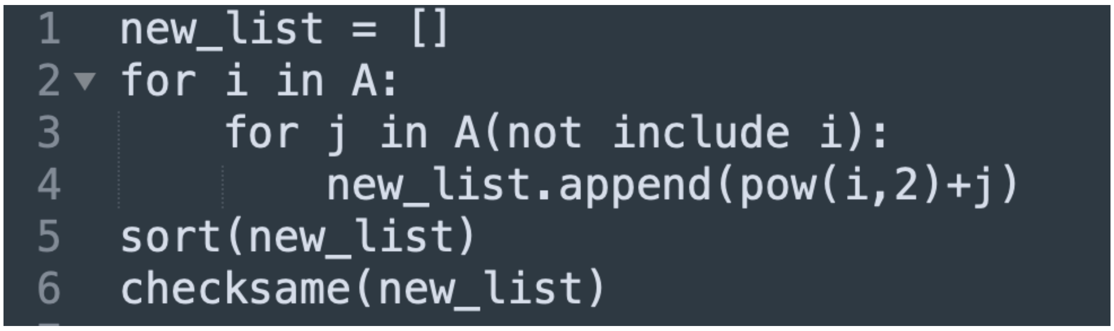
Name: Zanning Wang
zID: z5224151

1.

(a) the basic pseudo-code show below:



```
1  new_list = []
2  for i in A:
3      for j in A(not include i):
4          new_list.append(pow(i,2)+j)
5  sort(new_list)
6  checksame(new_list)
```

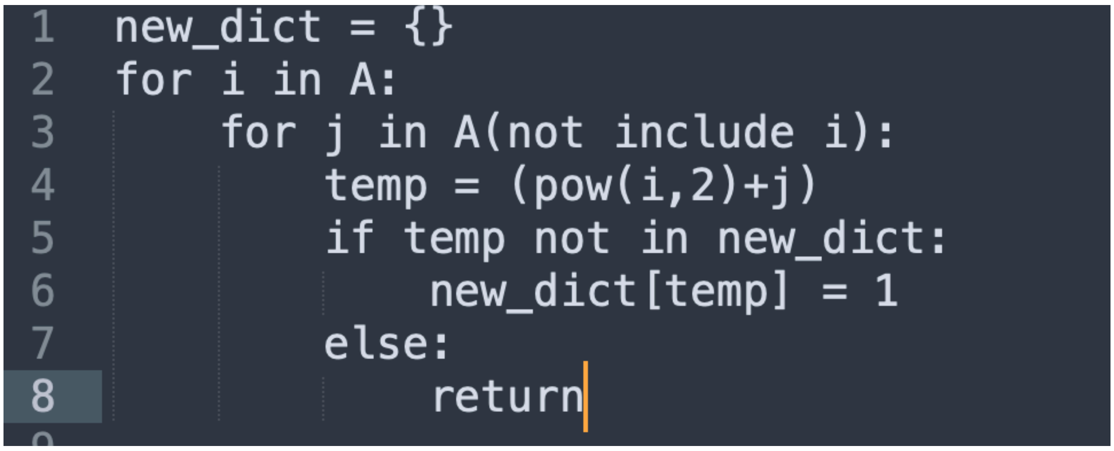
From line2-4, we calculate all possible x^2+y, and save them in a dict, for this step, the time complexity is $O(n*(n-1)) = O(n^2)$

From line5, we sort the list in order to help the next step we check if there exist two same numbers, because the length of current list is $O(n*(n-1)) = O(n^2)$, therefore the complexity of time for sorting this array is $O(n^2*\log(n^2)) = O(n^2* \log n)$

From the line5, we only need to traverse the array once to see if there are adjacent and identical numbers.

Therefore, the total time complexity is the most time-consuming step, which is step5, $O(n^2* \log n)$

(b)



```
1  new_dict = {}
2  for i in A:
3      for j in A(not include i):
4          temp = (pow(i,2)+j)
5          if temp not in new_dict:
6              new_dict[temp] = 1
7          else:
8              return
```

In order to decrease the time complexity, we may use the dictionary instead of the list, when we try to search one certain value, it only take O(n) time complexity.

From line2-4, it was same as the question (a), which take $O(n^2)$

From line 5-8, every time we calculate the x^2+y, we check if it exists in dictionary. If so, break the loop and return the result, else keep going, for this step, check element in dict only take O(n) time.

Therefore, the total time complexity is $O(n^2)$.