**Edgar Escobedo**

**Professor Diego Aguirre**

**TA: Anindita Nath**

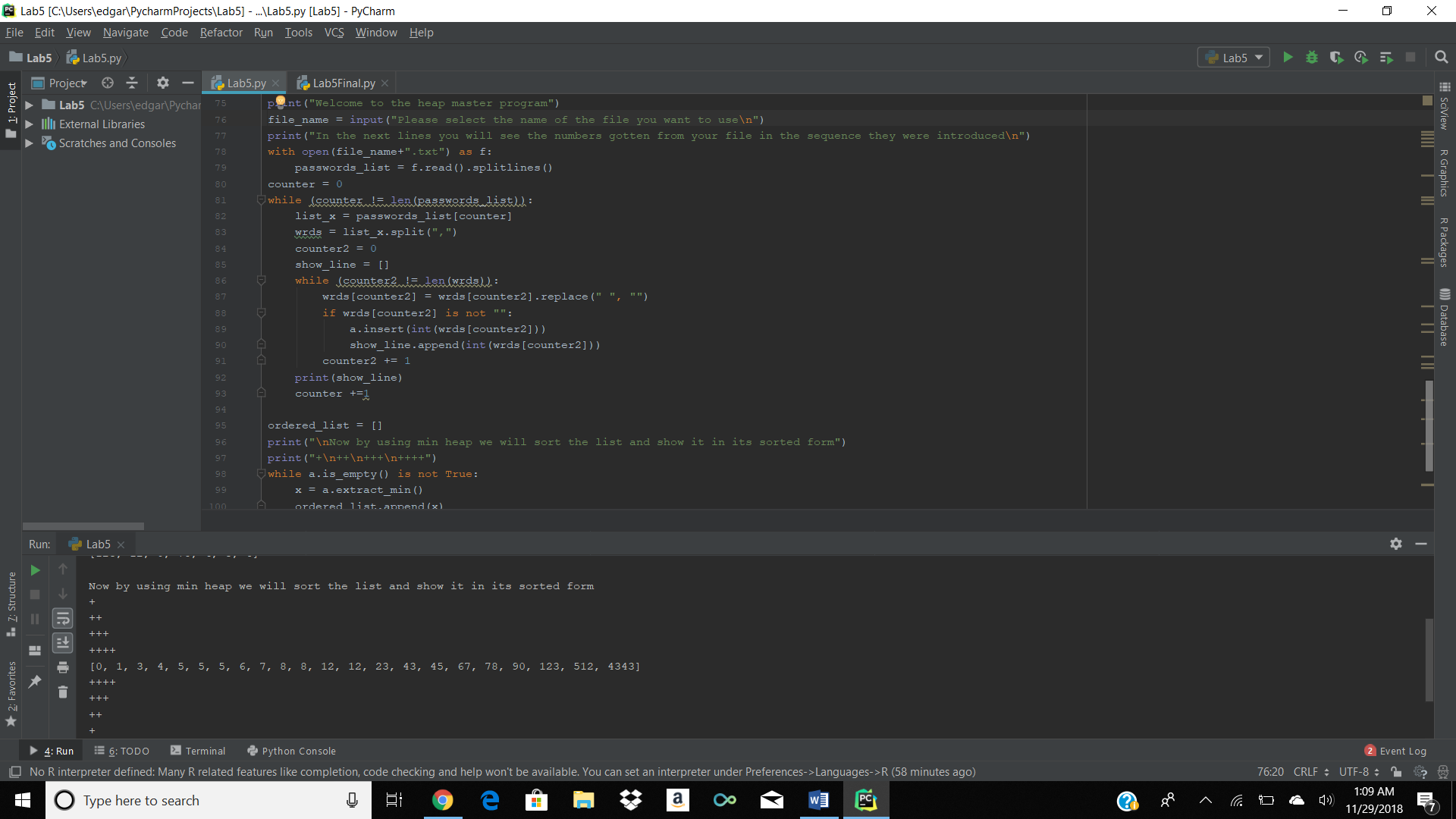
**CS 2302 Lab 5 Report**

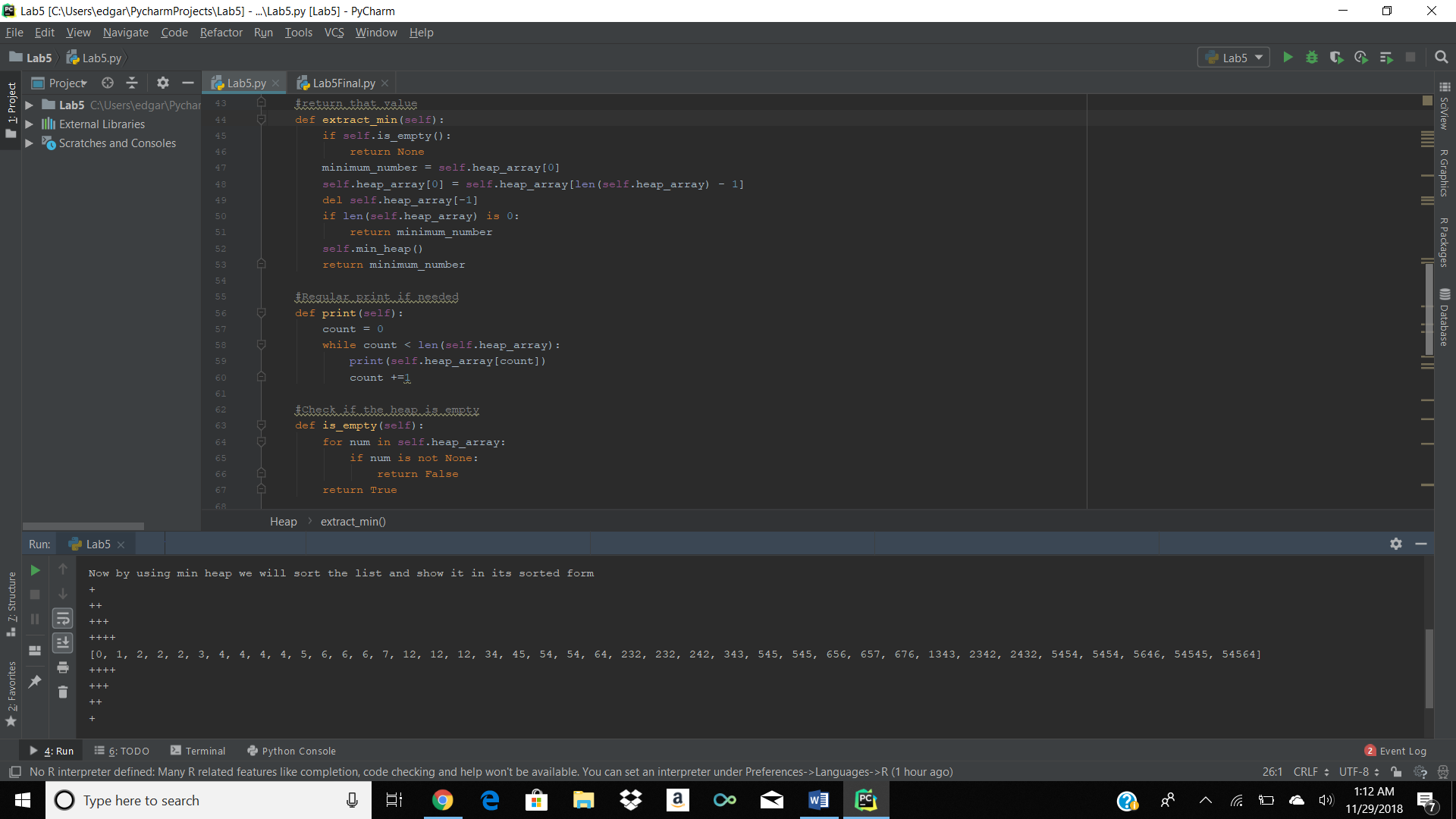
For lab 5 I decided to go for the option B, the task that we were required to do was to complete the implementation of the Min-Heap data structure, then by using this implementation we had to create the Heapsort method. After completing the data structures, we would test the methods by selection one of multiple options, in my case I selected the option of reading a file separated by commas and use the information of the file to sort it by the heapsort method.

The first step I took to solve this lab was to first create the heap by means of a list, where I would store the information. When this was completed, I had to continue with the method to add new information to the list, I would do so, by getting the value that I would add to the list and append it to the heap, then sending the current heap through the method of min\_heap. As expected, the next step was to create the min\_heap method which would help to maintain the properties of a Min-Heap. The method would return the same heap with no change if the length of the heap was 1, since it meant it already passed the Min-Heap conditions. For any heap of greater length, I would create a variable to the size of the heap minus 1 (to account for index starting at 0) and use the variable to traverse the heap from back to front, checking each index to its parent (we get the parent of the index by the formula (i -1//2)), if the index was smaller than its parent, we would swap the values at those positions to comply with the Min-Heap conditions.

Then I continued by creating an extract min method, which would extract the value at index 0 (the smallest value by the Min-Heap rules) and returning that value. When I stored the value at the index 0, I would change the value of the index 0, to the value at the last index of the heap, then call the min\_heap method to keep the Min-Heap structure. The last method that I created was the is empty method, which would traverse the whole heap, if at any point I would find any kind of information at the heap, it would return False, if I didn’t find anything in the list, I would return True. Finally, after all this method, I would read the file ignoring the commas and empty spaces, and just getting the numbers into the heap. Using a loop which would continue until the heap was empty, I would repeatedly call the min\_heap method and append the return value (value at index 0 of the current heap) into a new list, which would be the final sorted list.

In continuation the output of the lab will be shown to better illustrate the final product.





In conclusion this lab was very fun and educational to do, the knowledge that I gained about how heaps work, and creating my own min\_heap and heapsort data structures helped me to feel more confident with them. I believe that the way the lab was structured was very good because of the liberty that we were given, making you learn more by yourself, and not only trying to comply to certain requirements.

**Appendix:**

class Heap:  
 def \_\_init\_\_(self):  
 self.heap\_array = []  
  
  
 #Insertion of the new value and usage of the min heap with this new value to maintain the Min-Heap properties  
 def insert(self, val):  
 self.heap\_array.append(val)  
 self.min\_heap()  
  
  
 #Two base cases, if length of the heap is 1, it is already a min heap so just return, the second method will work for any size  
 #by checking all the indices, starting from the last one to its parent, being index -1 // 2. If the parent is greater than the  
 #index, we will swap them  
 def min\_heap(self):  
 if len(self.heap\_array) is 1:  
 return self  
 last\_element = len(self.heap\_array) - 1  
 while last\_element is not 0:  
 parent = (last\_element - 1) // 2  
 if self.heap\_array[last\_element] < self.heap\_array[parent]:  
 tmp = self.heap\_array[last\_element]  
 self.heap\_array[last\_element] = self.heap\_array[parent]  
 self.heap\_array[parent] = tmp  
 last\_element -= 1  
 return self  
  
  
 #Extract\_min will take the value at the first index, call min heap to main its properties, shrink the heap by 1 and  
 #return that value  
 def extract\_min(self):  
 if self.is\_empty():  
 return None  
 minimum\_number = self.heap\_array[0]  
 self.heap\_array[0] = self.heap\_array[len(self.heap\_array) - 1]  
 del self.heap\_array[-1]  
 if len(self.heap\_array) is 0:  
 return minimum\_number  
 self.min\_heap()  
 return minimum\_number  
  
 #Regular print if needed  
 def print(self):  
 count = 0  
 while count < len(self.heap\_array):  
 print(self.heap\_array[count])  
 count +=1  
  
 #Check if the heap is empty  
 def is\_empty(self):  
 for num in self.heap\_array:  
 if num is not None:  
 return False  
 return True  
  
a = Heap ()  
  
#In following lines the user will be asked the name of the file, the values of the files will be enter to the heap  
#and the sorted list will be shown  
  
  
print("Welcome to the heap master program")  
file\_name = input("Please select the name of the file you want to use\n")  
print("In the next lines you will see the numbers gotten from your file in the sequence they were introduced\n")  
with open(file\_name+".txt") as f:  
 passwords\_list = f.read().splitlines()  
counter = 0  
while (counter != len(passwords\_list)):  
 list\_x = passwords\_list[counter]  
 wrds = list\_x.split(",")  
 counter2 = 0  
 show\_line = []  
 while (counter2 != len(wrds)):  
 wrds[counter2] = wrds[counter2].replace(" ", "")  
 if wrds[counter2] is not "":  
 a.insert(int(wrds[counter2]))  
 show\_line.append(int(wrds[counter2]))  
 counter2 += 1  
 print(show\_line)  
 counter +=1  
  
ordered\_list = []  
print("\nNow by using min heap we will sort the list and show it in its sorted form")  
print("+\n++\n+++\n++++")  
while a.is\_empty() is not True:  
 x = a.extract\_min()  
 ordered\_list.append(x)  
print(ordered\_list)  
print("++++\n+++\n++\n+")  
print("\n Thank you")

I, Edgar Escobedo, certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.