**ISE4 Mini Project**

**Group 19**

**Hashcode 2020 Online Qualification Round**

**Team Members:**

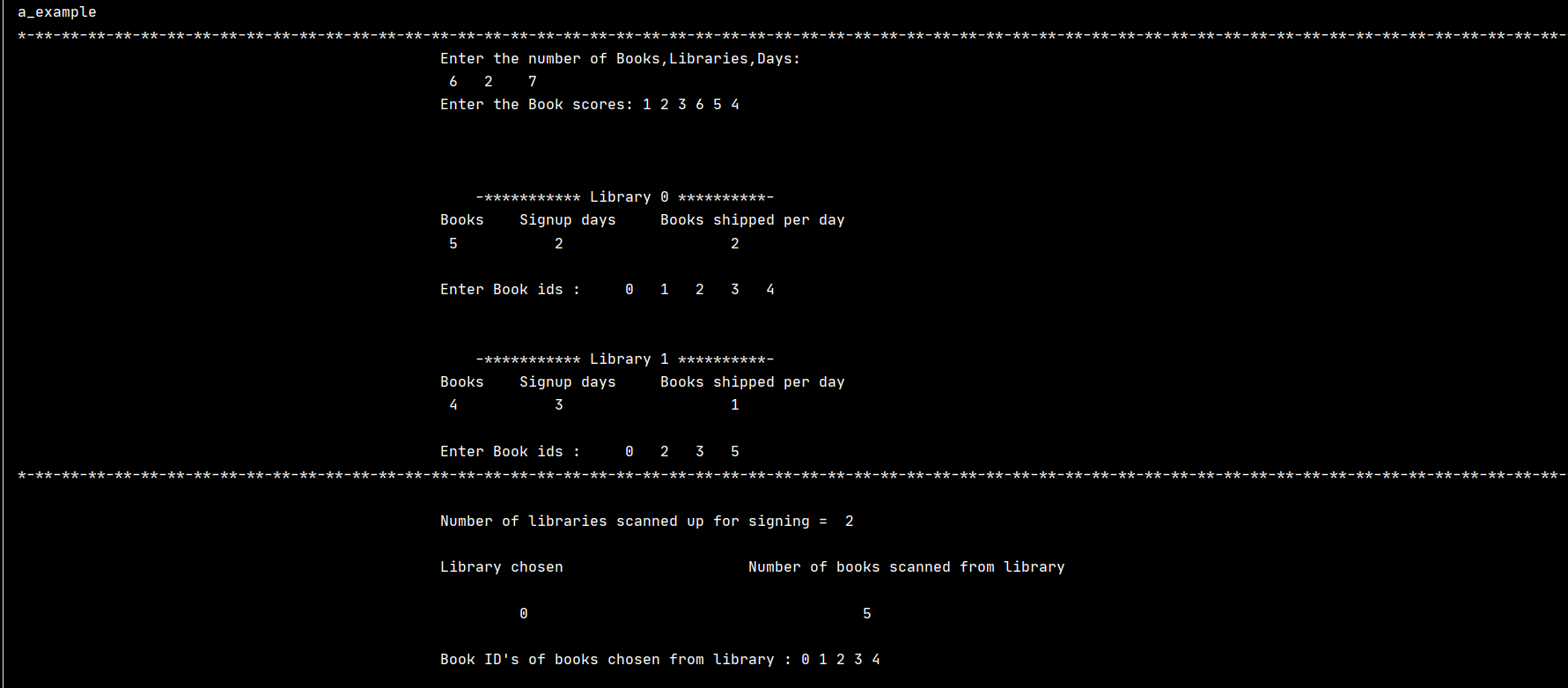
1. Aditi Bhosale (2019140009)
2. Anushka Agarwal (2019140002)
3. Shweta Chavan (2019140011)

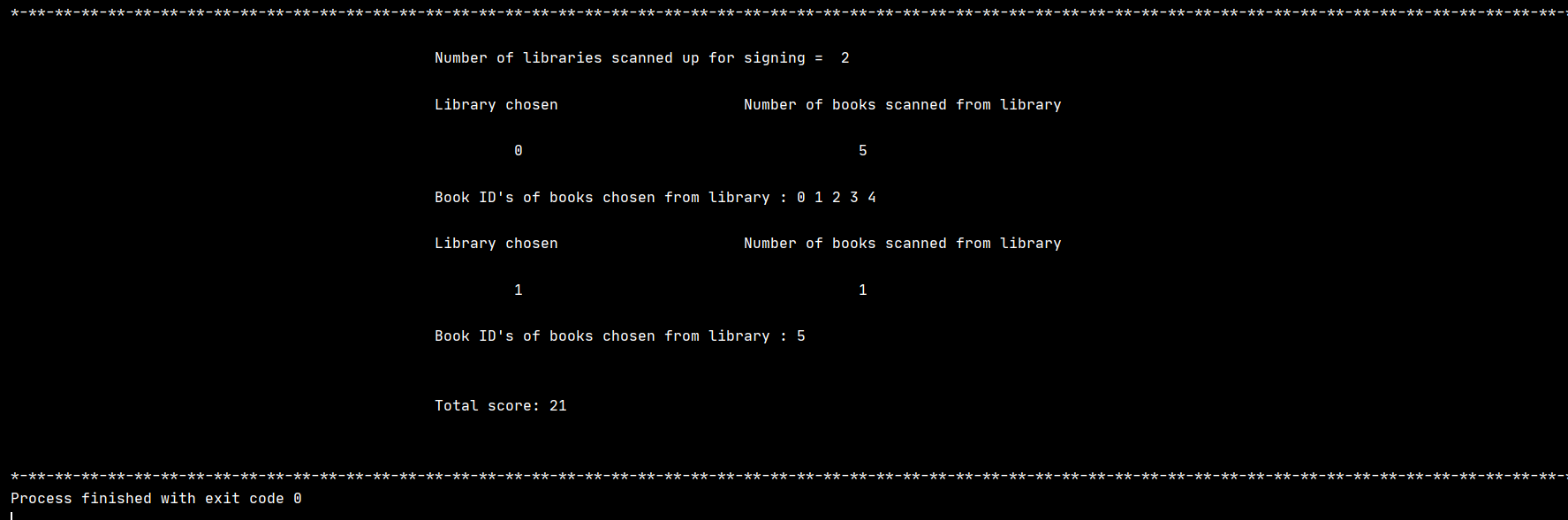
**Project Code:**

import numpy as np  
import operator  
import sys  
  
  
# signup score and library. We have to sort based on signup score, and hence item getter(0) since based on array1 we are sorting  
def parallel\_sort(array\_1**,** array\_2**,** ascending=False):  
 if not ascending:  
 return zip(\*sorted(zip(array\_1**,** array\_2)**,** key=operator.itemgetter(**0**))[::-**1**])  
 return zip(\*sorted(zip(array\_1**,** array\_2)**,** key=operator.itemgetter(**0**)))  
# zip is to create two arrays such that their keys are matching.  
  
class Library:  
  
 def \_\_init\_\_(self**,** id**,** book\_ids**,** signup\_days**,** max\_books\_scanned\_per\_day):  
 self.id = id  
 self.book\_ids = book\_ids  
 self.signup\_days = signup\_days  
 self.max\_books\_scanned\_per\_day = max\_books\_scanned\_per\_day  
  
 def get\_best\_book\_ids(self**,** start\_day=**0**):  
 global book\_scores**,** libraries\_signup\_days**,** libraries\_max\_books\_scanned\_per\_day**,** D**,** final\_books  
 available\_days = D - self.signup\_days - start\_day  
 #start day is different for different libraries, and available days also depends on previous libraries which had been there before.  
  
 available\_books = list(set(self.book\_ids) - set(final\_books))  
 current\_book\_scores = np.take(book\_scores**,** available\_books)  
 # returns array of same type with all the values of given indices. So here np.take will return an array which contains  
 # book scores of all available books  
  
 max\_num\_books = max(min(int(available\_days \* self.max\_books\_scanned\_per\_day)**,** len(available\_books))**, 0**)  
 # if available books is 1 and max scan possible is 2 then we have to scan 1 only hence the min thing.  
  
 if max\_num\_books == **0**:  
 return []  
  
 # get top k books (k=max\_num\_books)  
 ind = np.argpartition(current\_book\_scores**,** -max\_num\_books)[-max\_num\_books:]  
 # argpartition returns the array with index of the books in sorted order, and we have passed all values here  
  
 return np.take(available\_books**,** ind)  
  
 def get\_best\_books\_score(self**,** start\_day=**0**):  
 global book\_scores**,** libraries\_signup\_days**,** libraries\_max\_books\_scanned\_per\_day**,** D  
 available\_days = D - self.signup\_days - start\_day  
 max\_num\_books = max(min(int(available\_days \* self.max\_books\_scanned\_per\_day)**,** len(self.book\_ids))**, 0**)  
  
 current\_book\_scores = np.take(book\_scores**,** self.book\_ids)  
  
 # get top k books (k=max\_num\_books) argpartition returns the sorted index of all books since all values till max books #are passed one by one  
 ind = np.argpartition(current\_book\_scores**,** -max\_num\_books)[-max\_num\_books:]  
 best\_books\_scores = np.take(current\_book\_scores**,** ind)  
  
 return np.sum(best\_books\_scores)  
  
 # returns string representation of object  
 def \_\_repr\_\_(self):  
 return self.id.\_\_str\_\_()  
  
  
# sum of book scores of all book id's passed  
def sum\_book\_scores(book\_ids):  
 global book\_scores  
 return np.sum(np.take(book\_scores**,** list(book\_ids)))  
  
  
files = [  
 "a\_example"] # , "b\_read\_on", "c\_incunabula", "d\_tough\_choices", "e\_so\_many\_books", "f\_libraries\_of\_the\_world"]  
  
total\_score = **0**for file in files:  
  
 with open("inputs/" + file + ".txt"**,** "r") as f:  
 content = f.read().splitlines() # read each input lines differently  
 print(file) # prints the file name  
  
 for z in range(**60**):  
 print("\*-\*"**,** end='')  
 print("\n\t\t\t\t\t\t\t\t\t\t\t\tEnter the number of Books,Libraries,Days: ")  
 B**,** L**,** D = list(map(int**,** content[**0**].split(' ')))  
 print("\t\t\t\t\t\t\t\t\t\t\t\t"**,** B**,** "\t"**,** L**,** "\t "**,** D)  
 print("\t\t\t\t\t\t\t\t\t\t\t\tEnter the Book scores: "**,** end='')  
 book\_scores = list(map(int**,** content[**1**].split(' ')))  
 for x in range(len(book\_scores)):  
 print(book\_scores[x]**,** end=" ")  
 print("")  
 pos = **1** # library attributes : hence size of all below parameters are L  
 libraries\_num\_books = np.zeros(L) # number of books in lib 0,1,2 etc hence array of size L  
 libraries\_signup\_days = np.zeros(L) # signup days for the libraries hence array of size L  
 libraries\_max\_books\_scanned\_per\_day = np.zeros(L) # max books canned per day for the libraries  
  
 libraries = np.empty(L**,** dtype=Library) # array number of elements in row is L and data type is library  
 for i in range(L):  
 print("\n\n\n\t\t\t\t\t\t\t\t\t\t\t\t\t-\*\*\*\*\*\*\*\*\*\*\* Library"**,** i**,** "\*\*\*\*\*\*\*\*\*\*-"**,** " \n\t\t\t\t\t\t\t\t\t\t\t\tBooks \t Signup days \t Books shipped per day ")  
 pos += **1** n**,** t**,** m = list(map(int**,** content[pos].split(' ')))  
 libraries\_num\_books[i] = n  
 libraries\_signup\_days[i] = t  
 libraries\_max\_books\_scanned\_per\_day[i] = m  
 print("\t\t\t\t\t\t\t\t\t\t\t\t"**,** n**,** "\t\t\t"**,** t**,** "\t\t\t\t\t"**,** m)  
 pos += **1** print("\n\t\t\t\t\t\t\t\t\t\t\t\tEnter Book ids :"**,** end='')  
  
 book\_ids = np.asarray(list(map(int**,** content[pos].split(' '))))  
 for x in range(len(book\_ids)):  
 print("\t"**,** book\_ids[x]**,** end=" ")  
  
 libraries[i] = Library(i**,** book\_ids**,** t**,** m)  
  
 # Vectorization is used to speed up the Python code without using loop.  
 # Using such a function can help in minimizing the running time of code efficiently.  
 library\_book\_score\_counter = np.vectorize(lambda library: library.get\_best\_books\_score())  
 libraries\_scores = library\_book\_score\_counter(libraries)  
# library\_book\_score\_counter counts the score of the libraries,  
# we pass the array libraries and vectorize takes in each element of the array passed and gets the best book score  
# for it.The result is stored in libraries\_scores  
 heuristic\_score = np.vectorize(lambda book\_score**,** signup\_days: book\_score / signup\_days)  
 signup\_scores = heuristic\_score(libraries\_scores**,** libraries\_signup\_days)  
  
 signup\_scores**,** libraries\_sorted = parallel\_sort(signup\_scores**,** libraries)  
  
 final\_books = set()  
 with open(  
 "outputs/" + file + ".out"**,** 'w+') as f:  
  
 f.write(str(L) + "\n")  
 print()  
 for z in range(**60**):  
 print("\*-\*"**,** end='')  
  
 print('\n\n\t\t\t\t\t\t\t\t\t\t\t\tNumber of libraries scanned up for signing = '**,** str(L))  
 start\_day = **0** for i in range(L):  
  
 current\_library = libraries\_sorted[i]  
 chosen\_book\_ids = current\_library.get\_best\_book\_ids(start\_day)  
 final\_books.update(chosen\_book\_ids)  
 start\_day += current\_library.signup\_days  
  
 if len(chosen\_book\_ids) > **0**:  
 f.write(str(current\_library.id) + " " + str(len(chosen\_book\_ids)) + "\n")  
 f.write(str(' '.join(map(str**,** chosen\_book\_ids))) + "\n")  
 # print(str(current\_library.id) + " " + str(len(chosen\_book\_ids))  
 else:  
 f.write(str(current\_library.id) + " 1\n")  
 f.write(str(current\_library.book\_ids[**0**]) + "\n")  
  
 if len(chosen\_book\_ids) > **0**:  
 print("\n\t\t\t\t\t\t\t\t\t\t\t\tLibrary chosen \t\t\t\t\t Number of books scanned from library\n")  
 print(" \t\t\t\t\t\t\t\t\t\t\t\t\t\t"**,** str(current\_library.id) + " \t\t\t\t\t\t\t\t\t\t" + str(len(chosen\_book\_ids)) + "\n")  
 print("\t\t\t\t\t\t\t\t\t\t\t\tBook ID's of books chosen from library :"**,** end=" ")  
 print(str(' '.join(map(str**,** chosen\_book\_ids))))  
 # print(str(current\_library.id) + " " + str(len(chosen\_book\_ids))  
  
 else:  
 print(str(current\_library.id) + " 1\n")  
 print(str(current\_library.book\_ids[**0**]) + "\n")  
  
 # progress = 100 \* i / (2 \* L)  
 # sys.stdout.write("\rCreating output... (" + str(int(progress)) + " %)")  
  
 score = sum\_book\_scores(final\_books)  
 total\_score += score  
  
 # print("\r- Score:", score)  
  
print("")  
print("\n\t\t\t\t\t\t\t\t\t\t\t\tTotal score:"**,** total\_score**,** "\n\n")  
  
for z in range(**60**):  
 print("\*-\*"**,** end='')

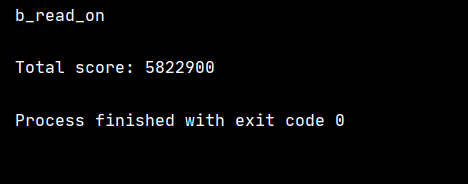
**Project Output:**

**Test Case 1:**

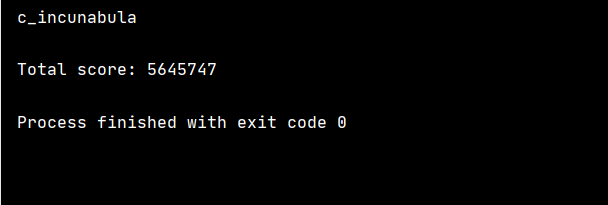




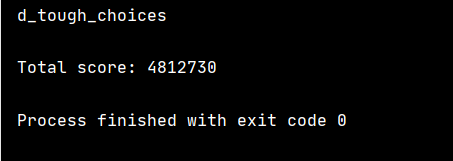
**Test Case 2:**



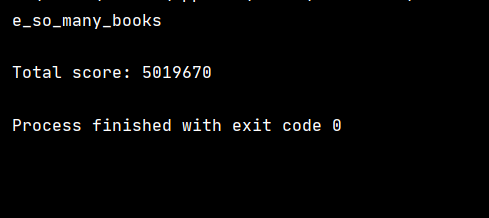
**Test Case 3:**



**Test Case 4:**

’

**Test Case 5:**



**Test Case 6:**

