REPORT ID: CMP4272W1L1

Self-Assessment: Please highlight where you think your report grade should be. Example below.

Criteria	Write simple algorithms using appropriate discrete data structures to solve computational problems (LO3)	Use appropriate methods to analyse the efficiency and correctness of algorithms (LO4)
Weight	**25%**	**25%**
0 – 29%	The algorithm does not solve an appropriate problem, or has serious errors. There is little or no discussion of how the algorithm works. No discrete data structure has been used, or the choice of data structure is inappropriate.	The analysis is limited and seriously flawed.
30 – 39%	The algorithm solves part of an appropriate problem. There may be substantial aspects of the problem which are not attempted or explained, or errors in the solution. The explanation is unclear or missing important details about how the algorithm works.	An attempt has been made to analyse an algorithm, but appropriate methods of analysis were not used, and the results of the analysis may be incorrect or meaningless.
40 – 49%	A rudimentary algorithm solving a basic problem. There may be some errors which could be corrected with further work. There is a limited discussion of how the algorithm works. The choice of data structure is inappropriate, or unjustified.	An attempt has been made to measure the running time of the algorithm for some inputs, but the methodology is unclear or the measurement may be inaccurate. There is a limited discussion of some other issues relating to efficiency. Analysis of the algorithm's correctness is vague, or not attempted.
50 – 59%	The algorithm solves an appropriate problem, though it may have minor errors or fail to account for special cases. There is an explanation of how the algorithm works. The choice of data structure may be inappropriate or poorly justified.	The running time of the algorithm has been measured accurately for an appropriate range of inputs, and the methodology has been explained. There is some discussion of other issues relating to efficiency. There is a basic or informal analysis of the algorithm's correctness.
60 – 69%	The algorithm correctly solves an appropriate problem. There is a clear explanation of how the algorithm works. At least one appropriate data structure has been used, and the choice has been adequately justified.	The efficiency of the algorithm has been accurately measured using an appropriate methodology, which has been explained. The measurements may include more than one metric. There is an analysis of the algorithm's correctness, which may specify pre- and post-conditions for part of the algorithm.
70 – 79%	The algorithm correctly solves a challenging problem. There is a clear explanation of how the algorithm works, and the explanation makes clear references to the relevant parts of the source code. Appropriate data structures have been used, and justification is given for each with reference to the specific problem.	The efficiency of the algorithm has been accurately measured using an appropriate methodology, with multiple metrics and a clear explanation. The asymptotic complexity of the algorithm is given. The efficiency may be compared with appropriate alternative algorithm(s). There is a formal analysis of the correctness of at least part of the algorithm.
80 - 90%	A well-designed algorithm which correctly solves a challenging problem. There is a clear, detailed explanation of how the algorithm works, with clear references to the relevant parts of the source code. Appropriate data structures have been used, and justification is given for each with reference to the specific problem.	The efficiency of the algorithm has been accurately measured using an appropriate methodology, with multiple metrics and a clear, detailed explanation. The asymptotic complexity of the algorithm is given. The efficiency has been compared with appropriate alternative algorithm(s). There is a detailed formal analysis of the correctness of the algorithm.
90 – 100%	An excellent algorithm written, explained and evaluated to the highest standards.	An excellent analysis of the efficiency, complexity and correctness of an algorithm, conducted and explained to the highest standards.

REPORT ID: CMP4272W1L1

BIRMINGHAM CITY University

Technical Report: **Muhamamd Abu-Baker**

Date: **10/05/2022**

Wordcount: **NUMBER Words**

Pagecount: **NUMBER Pages**

Confidential: **NO / [redacted] – INTERNAL ONLY/YES – X DEPARTMENT ONLY**

Contents

Theory and Implementations.......... 2

Insertion sort algorithm 2

Selection sort algortihm.......... 2

Linear search algorithm 3

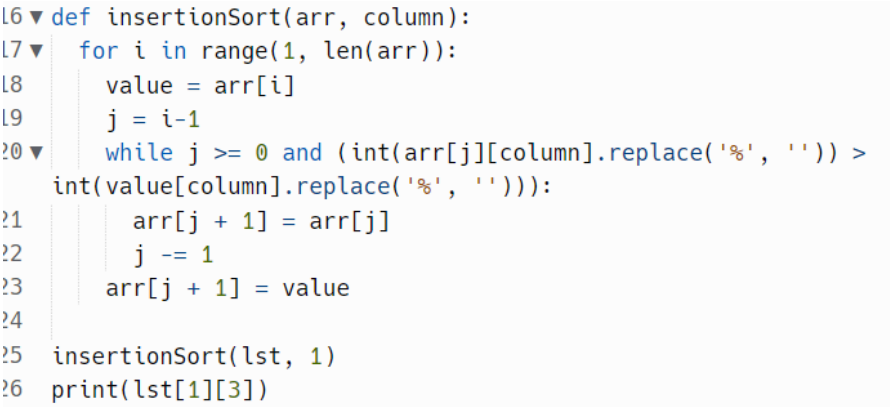
Quick sort algorithm 4

Binary search algorithm 5

Testing.......... 6

Theory and Implementations

Insertion sort algorithm



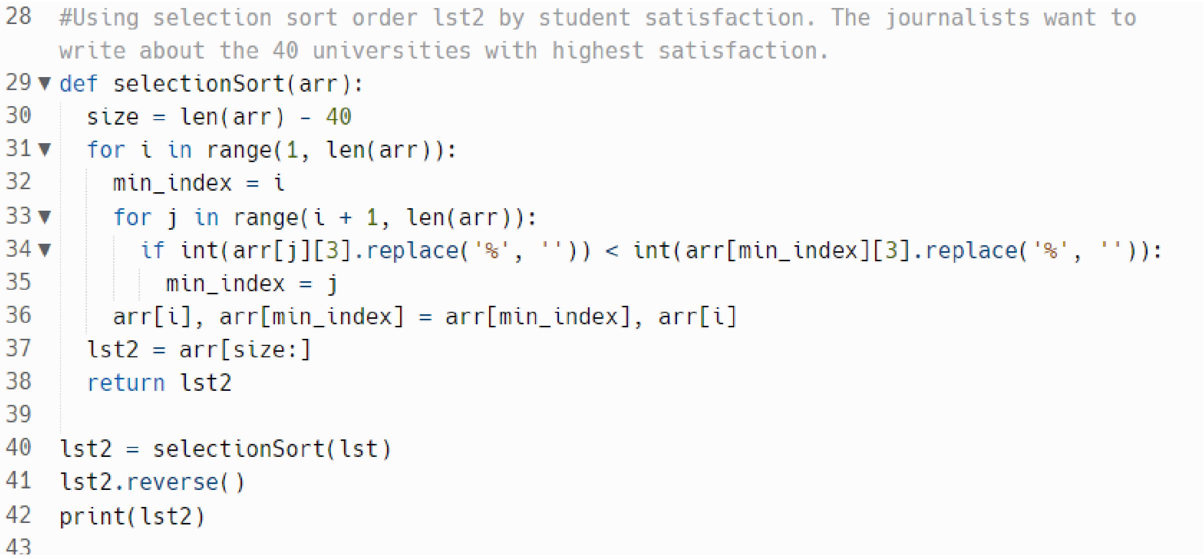
```python
def insertionSort(arr, column):
  for i in range(1, len(arr)):
    value = arr[i]
    j = i-1
    while j >= 0 and (int(arr[j][column].replace('%', '')) > int(value[column].replace('%', ''))):
      arr[j + 1] = arr[j]
      j -= 1
    arr[j + 1] = value

insertionSort(lst, 1)
print(lst[1][3])
```

An insertion sort is a method of inerstion sorting; it is used in this example to compare a specific key from each of the nested lists. The reason for choosing an insertion sort is we have a small array and it is more efficient for smaller data sets compared to a quick sort, merge sort or heap sort algorithm. Since our dataset is in a way sorted (by index as of now), the algoritm is adaptive.

The insertion sort function created is given two parameters, array and column, you then use a for loop in the range of 1 to the length of the given array. This looks for the length of the list values. You then need to create a variable for the value given as ‘i’ in the array to store the index value. Once complete, create another variable to start at the first index(j). Using a while loop to determine if j is greater than or equal to 0 as well as if the selected column and row in the array is an integer. This is done to compare if the value of the column is greater than the previous insertion. Inside the while loop, we state that in the array, index (j +) = equal to the value of j in the array. Thus subtracting it (j) by -1. Then exitting out fo the while loop, inside the for loop, assign array[j + 1] to the value to carry on the loop.

Lastly we call the function to sort the 2nd index in the array list and then print out the value at row 6, column 1.

Selection sort algortihm

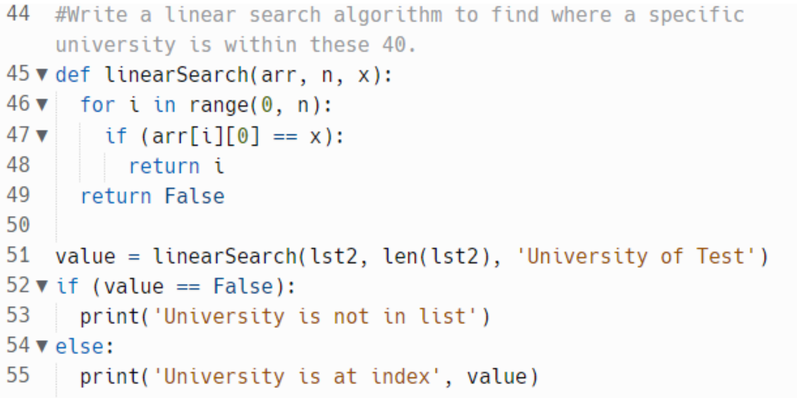


```python
#Using selection sort order lst2 by student satisfaction. The journalists want to write about the 40 universities with highest satisfaction.
def selectionSort(arr):
  size = len(arr) - 40
  for i in range(1, len(arr)):
    min_index = i
    for j in range(i + 1, len(arr)):
      if int(arr[j][3].replace('%', '')) < int(arr[min_index][3].replace('%', '')):
        min_index = j
    arr[i], arr[min_index] = arr[min_index], arr[i]
  lst2 = arr[size:]
  return lst2

lst2 = selectionSort(lst)
lst2.reverse()
print(lst2)
```

A selection sort is an algorithm used to sort an array/list. It is an inefficient way of sorting large amounts of data however, it's simplicity has many performance benefits. The algorithm is designed to separate a list/array in order to sort the list.

The selection sort function is given one parameter which is the array/list. Inside the function, we assign a variable the length of the array minus 40 to work out the top 40 universities. You then need a for loop in range of 1 to the length of the given array/list. Inside this for loop define a minimum index as 'i'. Once the variable has been assigned, create another for loop in range of 'i' and the length of the given array. Inside the for loop, I created an if statement to compare the values of the array at index j column 3, as our student satisfaction column is the third column. You then check if the minimum index in the given array is less than j's position in the array. We then assign j to the minimum index. Exiting out of this current if statement, you make the current array index equal to the next minimum index and the current mininmun index of the array to another index of the given array. Exiting out into the first for loop, im assigning lst2 to the given array slicing it to start from the rest of the array and then returning it. Outside of the function, we create assign lst2 as a selction sorted lst and then reverse it to receive the top 40 universities when we print lst2.

Linear search algorithm

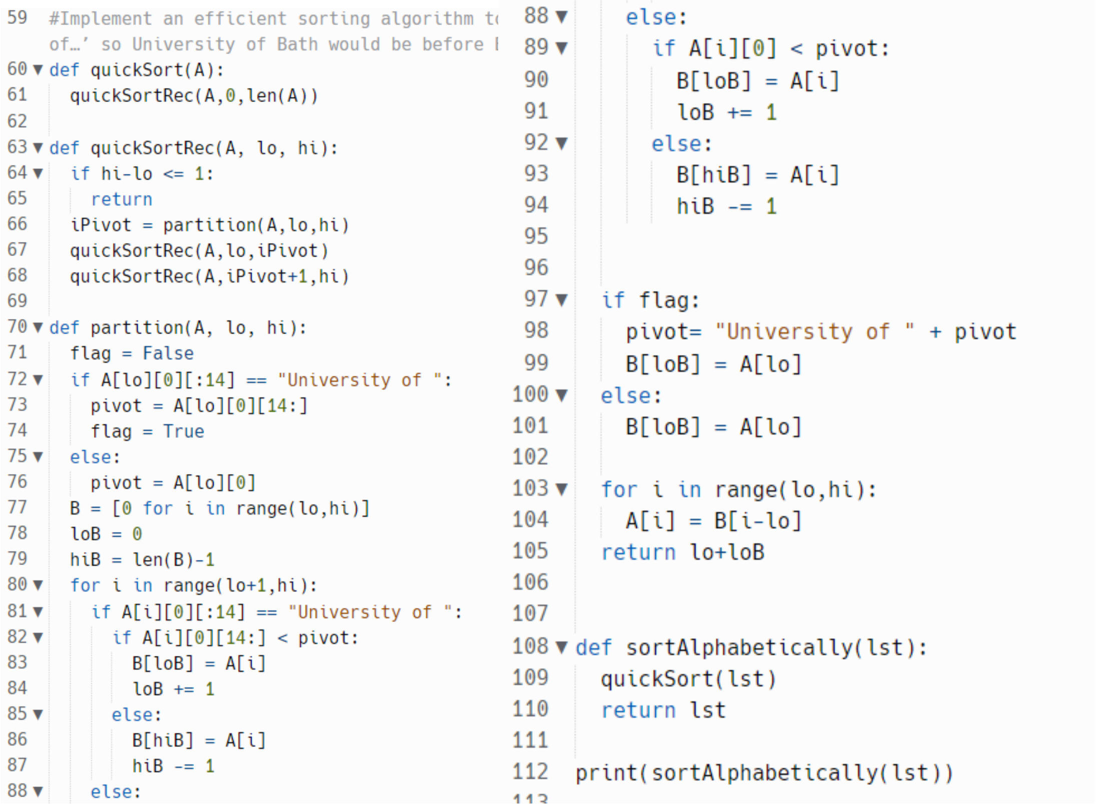


```python
#Write a linear search algorithm to find where a specific university is within these 40.
def linearSearch(arr, n, x):
  for i in range(0, n):
    if (arr[i][0] == x):
      return i
  return False

value = linearSearch(lst2, len(lst2), 'University of Test')
if (value == False):
  print('University is not in list')
else:
  print('University is at index', value)
```

A linear sort algorithm is used to find an element within a given array, sequentially checking the entire list for an element until it has been found.

We start off by creating a linear search function with the parameters array, n and x, to locate where a specific university is in the list of the top 40. Therefore, creating a for loop for i in range of 0 and n, inside this for loop we need an if statement to compare if the value i at index 0 in the given array is equal to the x parameter. If it is, we return i, if not then we exit into the previous for loop and return False. We then need a variable to define the value and use the linear search function to find with the arguments lst2 for array, the length of lst2 and a test value to find it there is a university with that name that exists in the sorted array. Lastly printing off the result if it is in the array or giving an error statement if it is not in the array.

Quick sort algorithm

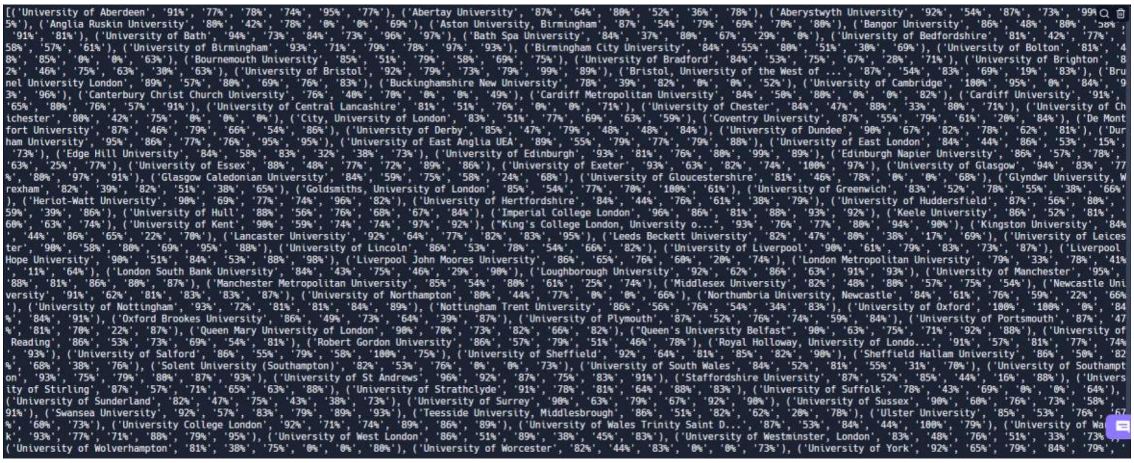


```
#Implement an efficient sorting algorithm t
of…' so University of Bath would be before I
def quickSort(A):
    quickSortRec(A,0,len(A))

def quickSortRec(A, lo, hi):
    if hi-lo <= 1:
        return
    iPivot = partition(A,lo,hi)
    quickSortRec(A,lo,iPivot)
    quickSortRec(A,iPivot+1,hi)

def partition(A, lo, hi):
    flag = False
    if A[lo][0][:14] == "University of ":
        pivot = A[lo][0][14:]
        flag = True
    else:
        pivot = A[lo][0]
    B = [0 for i in range(lo,hi)]
    loB = 0
    hiB = len(B)-1
    for i in range(lo+1,hi):
        if A[i][0][:14] == "University of ":
            if A[i][0][14:] < pivot:
                B[loB] = A[i]
                loB += 1
            else:
                B[hiB] = A[i]
                hiB -= 1
        else:
            if A[i][0] < pivot:
                B[loB] = A[i]
                loB += 1
            else:
                B[hiB] = A[i]
                hiB -= 1


    if flag:
        pivot= "University of " + pivot
        B[loB] = A[lo]
    else:
        B[loB] = A[lo]

    for i in range(lo,hi):
        A[i] = B[i-lo]
    return lo+loB


def sortAlphabetically(lst):
    quickSort(lst)
    return lst

print(sortAlphabetically(lst))
```

A quick sort algorithm is referred to as a divide and conquer algorithm, utilising a pivot element from a given array and sectionalisation of the other elements into two sub arrays.

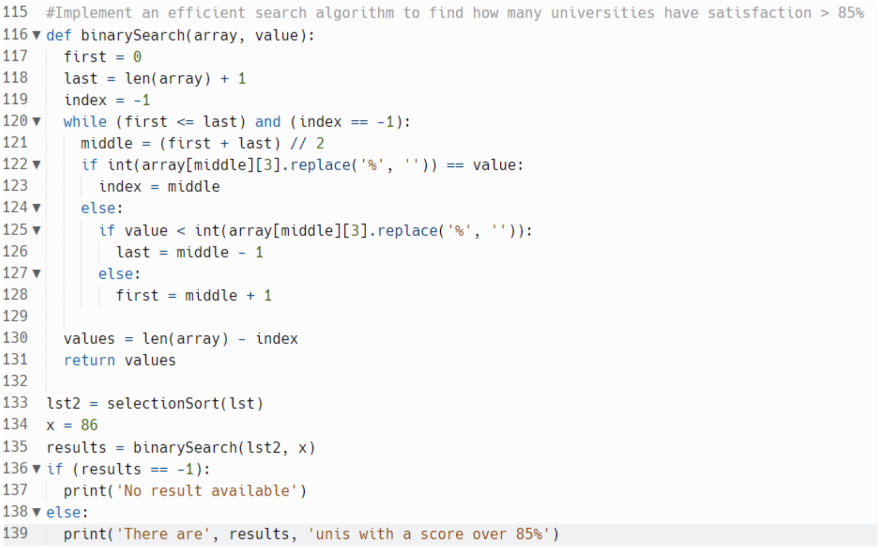
We need to use a quick sort algorithm to sort the names of each universities, without including the 'university of' bit. To do this, we create multiple functions. The first function will be quick sorting the given array with a parameter in the function. It will be using the next quick sort function that will be recursive and has three given parameters to help determine the pivot points in the array. The quick sorting recursive function has the parameters array, low and high. Using an if statement, if high minus low is less than and equal to 1 it return back to the function. We then need to assign variables to find the index of pivot, and call the recursive quick sort function with the new index pivot variable.

In summary, we use quick sort to look at the first 14 characters of the university name index, sorting it alphabetically and if it contains 'University of ', it sorts it from the name/word after 'University of '.



```
[('University of Aberdeen', '91%', '77%', '78%', '74%', '95%', '77%'), ('Abertay University', '87%', '64%', '80%', '52%', '36%', '78%'), ('Aberystwyth University', '92%', '54%', '87%', '73%', '99%
5%'), ('Anglia Ruskin University', '80%', '42%', '78%', '0%', '0%', '69%'), ('Aston University, Birmingham', '87%', '54%', '79%', '69%', '70%', '80%'), ('Bangor University', '86%', '48%', '80%', '58%',
 '91%', '81%'), ('University of Bath', '94%', '73%', '84%', '73%', '96%', '97%'), ('Bath Spa University', '84%', '37%', '80%', '67%', '29%', '0%'), ('University of Bedfordshire', '81%', '42%', '77%', '
58%', '57%', '61%'), ('University of Birmingham', '93%', '71%', '79%', '78%', '97%', '93%'), ('Birmingham City University', '84%', '55%', '80%', '51%', '30%', '69%'), ('University of Bolton', '81%', '4
8%', '85%', '0%', '0%', '63%'), ('Bournemouth University', '85%', '51%', '79%', '58%', '69%', '75%'), ('University of Bradford', '84%', '53%', '75%', '67%', '28%', '71%'), ('University of Brighton', '8
2%', '46%', '75%', '63%', '30%', '63%'), ('University of Bristol', '92%', '79%', '73%', '79%', '99%', '89%'), ('Bristol, University of the West of ...', '87%', '54%', '83%', '69%', '19%', '83%'), ('Bru
nel University London', '89%', '57%', '80%', '69%', '76%', '83%'), ('Buckinghamshire New University', '78%', '39%', '82%', '0%', '0%', '52%'), ('University of Cambridge', '100%', '95%', '0%', '84%', '9
3%', '96%'), ('Canterbury Christ Church University', '76%', '40%', '70%', '0%', '0%', '49%'), ('Cardiff Metropolitan University', '84%', '50%', '80%', '0%', '0%', '82%'), ('Cardiff University', '91%',
'65%', '80%', '76%', '57%', '91%'), ('University of Central Lancashire', '81%', '51%', '76%', '0%', '0%', '71%'), ('University of Chester', '84%', '47%', '88%', '33%', '80%', '71%'), ('University of Ch
ichester', '80%', '42%', '75%', '0%', '0%', '0%'), ('City, University of London', '83%', '51%', '77%', '69%', '63%', '59%'), ('Coventry University', '87%', '55%', '79%', '61%', '20%', '84%'), ('De Mont
fort University', '87%', '46%', '79%', '66%', '54%', '86%'), ('University of Derby', '85%', '47%', '79%', '48%', '48%', '84%'), ('University of Dundee', '90%', '67%', '82%', '78%', '62%', '81%'), ('Dur
ham University', '95%', '86%', '77%', '76%', '95%', '95%'), ('University of East Anglia UEA', '89%', '55%', '79%', '77%', '79%', '88%'), ('University of East London', '84%', '44%', '86%', '53%', '15%',
 '73%'), ('Edge Hill University', '84%', '58%', '83%', '32%', '38%', '73%'), ('University of Edinburgh', '93%', '81%', '76%', '80%', '99%', '89%'), ('Edinburgh Napier University', '86%', '57%', '78%',
'63%', '25%', '77%'), ('University of Essex', '88%', '48%', '77%', '72%', '89%', '86%'), ('University of Exeter', '93%', '63%', '82%', '74%', '100%', '97%'), ('University of Glasgow', '94%', '83%', '77
%', '80%', '97%', '91%'), ('Glasgow Caledonian University', '84%', '59%', '75%', '58%', '24%', '68%'), ('University of Gloucestershire', '81%', '46%', '78%', '0%', '0%', '68%'), ('Glyndwr University, W
rexham', '82%', '39%', '82%', '51%', '38%', '65%'), ('Goldsmiths, University of London', '85%', '54%', '77%', '70%', '100%', '61%'), ('University of Greenwich', '83%', '52%', '78%', '55%', '38%', '66%'
), ('Heriot-Watt University', '90%', '69%', '77%', '74%', '96%', '82%'), ('University of Hertfordshire', '84%', '44%', '76%', '61%', '38%', '79%'), ('University of Huddersfield', '87%', '56%', '80%', '
59%', '39%', '86%'), ('University of Hull', '88%', '56%', '76%', '68%', '67%', '84%'), ('Imperial College London', '96%', '86%', '81%', '88%', '93%', '92%'), ('Keele University', '86%', '52%', '81%', '
60%', '63%', '74%'), ('University of Kent', '90%', '59%', '74%', '74%', '97%', '92%'), ("King's College London, University o...", '93%', '76%', '77%', '80%', '94%', '90%'), ('Kingston University', '84%
', '44%', '86%', '65%', '22%', '70%'), ('Lancaster University', '92%', '64%', '77%', '82%', '83%', '95%'), ('Leeds Beckett University', '82%', '47%', '80%', '38%', '17%', '69%'), ('University of Leices
ter', '90%', '58%', '80%', '69%', '95%', '88%'), ('University of Lincoln', '86%', '53%', '78%', '54%', '66%', '82%'), ('University of Liverpool', '90%', '61%', '79%', '83%', '73%', '87%'), ('Liverpool
Hope University', '90%', '51%', '84%', '53%', '88%', '98%'), ('Liverpool John Moores University', '86%', '65%', '76%', '60%', '20%', '74%'), ('London Metropolitan University', '79%', '33%', '78%', '41%
', '11%', '64%'), ('London South Bank University', '84%', '43%', '75%', '46%', '29%', '90%'), ('Loughborough University', '92%', '62%', '86%', '63%', '91%', '93%'), ('University of Manchester', '95%',
'88%', '81%', '86%', '80%', '87%'), ('Manchester Metropolitan University', '85%', '54%', '80%', '61%', '25%', '74%'), ('Middlesex University', '82%', '48%', '80%', '57%', '75%', '54%'), ('Newcastle Uni
versity', '91%', '62%', '81%', '83%', '83%', '87%'), ('University of Northampton', '80%', '44%', '77%', '0%', '0%', '66%'), ('Northumbria University, Newcastle', '84%', '61%', '76%', '59%', '22%', '66%
'), ('University of Nottingham', '93%', '72%', '81%', '81%', '84%', '89%'), ('Nottingham Trent University', '86%', '56%', '76%', '54%', '34%', '83%'), ('University of Oxford', '100%', '100%', '0%', '84
%', '84%', '91%'), ('Oxford Brookes University', '86%', '49%', '73%', '64%', '39%', '87%'), ('University of Plymouth', '87%', '52%', '76%', '74%', '59%', '84%'), ('University of Portsmouth', '87%', '47
%', '81%', '70%', '22%', '87%'), ('Queen Mary University of London', '90%', '70%', '73%', '82%', '66%', '82%'), ("Queen's University Belfast", '90%', '63%', '75%', '71%', '92%', '88%'), ('University of
 Reading', '86%', '53%', '73%', '69%', '54%', '81%'), ('Robert Gordon University', '86%', '57%', '79%', '51%', '46%', '78%'), ('Royal Holloway, University of Londo...', '91%', '57%', '81%', '77%', '74%
', '93%'), ('University of Salford', '86%', '55%', '79%', '58%', '100%', '75%'), ('University of Sheffield', '92%', '64%', '81%', '85%', '82%', '90%'), ('Sheffield Hallam University', '86%', '50%', '82
%', '68%', '38%', '76%'), ('Solent University (Southampton)', '82%', '53%', '76%', '0%', '0%', '73%'), ('University of South Wales', '84%', '52%', '81%', '55%', '31%', '70%'), ('University of Southampt
on', '93%', '75%', '79%', '80%', '87%', '93%'), ('University of St Andrews', '96%', '92%', '87%', '75%', '83%', '91%'), ('Staffordshire University', '87%', '52%', '85%', '44%', '16%', '88%'), ('Univers
ity of Stirling', '87%', '57%', '71%', '65%', '63%', '88%'), ('University of Strathclyde', '91%', '78%', '81%', '64%', '88%', '83%'), ('University of Suffolk', '78%', '43%', '69%', '0%', '0%', '64%'),
('University of Sunderland', '82%', '47%', '75%', '43%', '38%', '73%'), ('University of Surrey', '90%', '63%', '79%', '67%', '92%', '90%'), ('University of Sussex', '90%', '60%', '76%', '73%', '58%', '
91%'), ('Swansea University', '92%', '57%', '83%', '79%', '89%', '93%'), ('Teesside University, Middlesbrough', '86%', '51%', '82%', '62%', '20%', '78%'), ('Ulster University', '85%', '53%', '76%', '67
%', '60%', '73%'), ('University College London', '92%', '71%', '74%', '89%', '86%', '89%'), ('University of Wales Trinity Saint D...', '87%', '53%', '84%', '44%', '100%', '79%'), ('University of Wa
k', '93%', '77%', '71%', '88%', '79%', '95%'), ('University of West London', '86%', '51%', '89%', '38%', '45%', '83%'), ('University of Westminster, London', '83%', '48%', '76%', '51%', '33%', '73%
('University of Wolverhampton', '81%', '38%', '75%', '0%', '0%', '80%'), ('University of Worcester', '82%', '44%', '83%', '0%', '0%', '73%'), ('University of York', '92%', '65%', '79%', '84%', '79%', '
```

Binary search algorithm



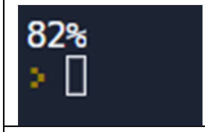
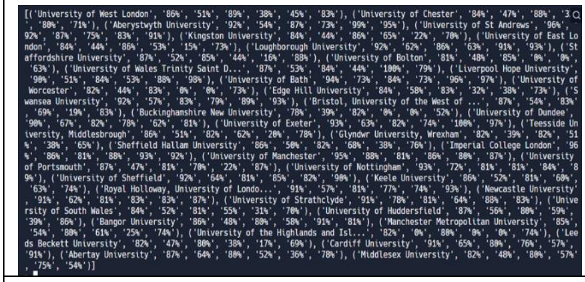
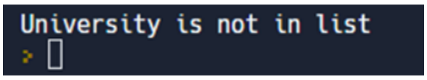
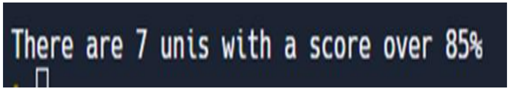
```
#Implement an efficient search algorithm to find how many universities have satisfaction > 85%
def binarySearch(array, value):
  first = 0
  last = len(array) + 1
  index = -1
  while (first <= last) and (index == -1):
    middle = (first + last) // 2
    if int(array[middle][3].replace('%', '')) == value:
      index = middle
    else:
      if value < int(array[middle][3].replace('%', '')):
        last = middle - 1
      else:
        first = middle + 1

  values = len(array) - index
  return values

lst2 = selectionSort(lst)
x = 86
results = binarySearch(lst2, x)
if (results == -1):
  print('No result available')
else:
  print('There are', results, 'unis with a score over 85%')
```

Binary search is an efficient logarithmic search which discovers the target value’s index within a sorted array. We need to use an efficient search algorithm to look for the amount of universitys with a score over 85%, therefore I chose to use a binary search algorithm.

Creating a function with the parameters array and value will help to locate the target results. Then I create variables for the first index (equal to 0), last index (equal to the length of the given array + 1) and an index value of -1. Then we use a while loop with the arguments (first is less than and equal to last) AND index is equal to 1. Then indenting to define the middle variable as being the sum of the first and last variable using floor division by 2.

In summary, the binary search locates all values of 86% and above in the column student satisfaction.

Testing

Test name	Efficiency	Correctness
insertion sort  82%	The insertion sort was performance was quick and produced a value instantly	In this case, I believe insertion sort was a good choice for sorting algorithm for the required task
selection sort  [('University of West London', '86%', '51%', '89%', '38%', '45%', '83%'), ('University of Chester', '84%', '47%', '88%', '3 ', '80%', '71%'), ('Aberystwyth University', '92%', '54%', '87%', '73%', '99%', '95%'), ('University of St Andrews', '96%', 92%', '87%', '75%', '83%', '91%'), ('Kingston University', '84%', '44%', '86%', '65%', '22%', '70%'), ('University of East Lo ndon', '84%', '44%', '86%', '53%', '15%', '73%'), ('Loughborough University', '92%', '62%', '86%', '63%', '91%', '93%'), ('St affordshire University', '87%', '52%', '85%', '44%', '16%', '88%'), ('University of Bolton', '81%', '48%', '85%', '0%', '0%', '63%'), ('University of Wales Trinity Saint D...', '87%', '53%', '84%', '44%', '100%', '79%'), ('Liverpool Hope University', '90%', '51%', '84%', '53%', '88%', '98%'), ('University of Bath', '94%', '73%', '84%', '73%', '96%', '97%'), ('University of Worcester', '82%', '44%', '83%', '0%', '0%', '73%'), ('Edge Hill University', '84%', '58%', '83%', '32%', '38%', '73%'), ('S wansea University', '92%', '57%', '83%', '79%', '89%', '93%'), ('Bristol, University of the West of ...', '87%', '54%', '83%' , '69%', '19%', '83%'), ('Buckinghamshire New University', '78%', '39%', '82%', '0%', '0%', '52%'), ('University of Dundee', '90%', '67%', '82%', '78%', '62%', '81%'), ('University of Exeter', '93%', '63%', '82%', '74%', '100%', '97%'), ('Teesside Un iversity, Middlesbrough', '86%', '51%', '82%', '62%', '20%', '78%'), ('Glyndwr University, Wrexham', '82%', '39%', '82%', '51 %', '38%', '65%'), ('Sheffield Hallam University', '86%', '50%', '82%', '68%', '38%', '76%'), ('Imperial College London', '96 %', '86%', '81%', '88%', '93%', '92%'), ('University of Manchester', '95%', '88%', '81%', '86%', '80%', '87%'), ('University of Portsmouth', '87%', '47%', '81%', '70%', '22%', '87%'), ('University of Nottingham', '93%', '72%', '81%', '81%', '84%', '8 9%'), ('University of Sheffield', '92%', '64%', '81%', '85%', '82%', '90%'), ('Keele University', '86%', '52%', '81%', '60%', '63%', '74%'), ('Royal Holloway, University of Londo...', '91%', '57%', '81%', '77%', '74%', '93%'), ('Newcastle University' , '91%', '62%', '81%', '83%', '83%', '87%'), ('University of Strathclyde', '91%', '78%', '81%', '64%', '88%', '83%'), ('Unive rsity of South Wales', '84%', '52%', '81%', '55%', '31%', '70%'), ('University of Huddersfield', '87%', '56%', '80%', '59%', '39%', '86%'), ('Bangor University', '86%', '48%', '80%', '58%', '91%', '81%'), ('Manchester Metropolitan University', '85%', '54%', '80%', '61%', '25%', '74%'), ('University of the Highlands and Isl...', '82%', '0%', '80%', '0%', '0%', '74%'), ('Lee ds Beckett University', '82%', '47%', '80%', '38%', '17%', '69%'), ('Cardiff University', '91%', '65%', '80%', '76%', '57%', '91%'), ('Abertay University', '87%', '64%', '80%', '52%', '36%', '78%'), ('Middlesex University', '82%', '48%', '80%', '57%' , '75%', '54%')]	The algorithm runs fast and productively, producing the results instantly as it deals with a minor quantity of data.	Since the algorithm is a simple one, it is effective in sorting the columns correctly with no issues. However if the given dataset was larger, the selction sort may have malfunctioned by sorting incorrectly.
Linear search  University is not in list	Linear searches are more efficient the smaller the data sets. It will either take too long or will become imprecise.	Choosing this algorithm was correct as it deals with a small dataset therefore, providing a faster more accurate result.
Binary search  There are 7 unis with a score over 85%	A binary search is an resourceful algorithm to search for an index’s value within a given array.	This algorithm was chosen because it is fast at finding the specified value as well as being accurate.