Queensland University of Technology
CAB301: Data Structures and Algorithms

Development of a Software Application for a Tool Library (Technical Report)

Due: 11:59 pm Monday 24th May (Week 12)

Name: Jol

ID Number: n10755756

Table of Contents

Table of Contents 2
1.0 Introduction 3
2.0 Design and Analysis of Algorithms 4
2.1 - Algorithm 4
2.2 - Heap-Sort Pseudocode 4
2.2 - Theoretical Analysis 4
2.3 - Empirical Analysis 5
2.4 - Summary Justification of Empirical and Theoretical Analysis 6
3.0 - Summary of Algorithm 6
4.0 Software Test Plans and Test Results 7
References 13

1.0 Introduction

I've been tasked to develop a tool library software application in Visual Studio that manages information about tools and members of the tool library. The solution I've implemented from the design specifications implements a Heap Sort and Binary Tree Search Traversal Sorting, moreover, my solution has a members menu where they can: borrow a tool, return a tool, display a list of tools from a certain category, display tools they're currently borrowing or display the top three most frequent borrowed tools, and a staff menu where staff can: add new tools, increase/decrease the quantity of certain tools, register new members, remove members, and find a members contact number.

Furthermore, numerous deviations were encountered throughout the development process, these consisted of poorly constructed, and designed interfaces that were provided. For example, interfaces containing iTool or iMember had to be modified to Tool/Member to increase its accessibility and usability in the ToolLibraySystem, Member, and in many other classes. Another deviation created this project an appalling development was the abysmal, and poorly written specifications as I and many others had to submit a number of inquiries in clarifying certain components of both specifications and interfaces. In addition, if given the ability to change certain aspects of this project, it would most definitely be the horribly designed interfaces because I would've made it more simplistic instead of spending hours on end to find work arounds.

Unfortunately, certain aspects of my program aren't fully functional, these consist of adding/removing a tool, pieces of a tool, a member, and others, moreover, the program allows input, but the tools, tool quantity, members, etc classes don't update. Aside from these deviations, the Binary Search Tree Traversal Sorting algorithm was utilised in handling and retrieving members from the MemberCollection class, and the heap sort algorithm was used in sorting the top 3 most frequently borrowed tools.

2.0 Design and Analysis of Algorithms

2.1 - Algorithm

The Heap Sort algorithm is a much more efficient variant of the selection sort, this algorithm functions by determining the largest or smallest element of a list, placing the element at the end or beginning of the list, and then repeats this task for every other element in the list, but accomplishes the sorting efficiently by utilising the data structure called heap, which's a special type of a binary tree. Furthermore, once the lists data is transformed into a heap, the root node is known to either be the largest or smallest element, also when the root is removed and placed at the end of the list, heap is then rearranged so that the largest element remaining is moved to the root. Furthermore, the heap sort algorithm was chosen to display the top three most frequently borrowed tools as it has a worst/best/average-time complexity of O (n log n).

2.2 - Heap-Sort Pseudocode

ALGORITHM Tool[] *HeapBottomUp*(Tool[] h[1..n])
//Constructing a heap from Tool class by the bottom-up algorithm
//Input: An array H[1..n] of orderable items
//Output: A heap H[1..n]
for i ← [n/2] **downto** 1 **do**
    k ← I; v ←h[k]
    heap ← false
    while not heap **and** 2*k ≤ n **do**
    j ← 2*k
    if j < n *//two children*
        if H[j] < H[j+1] j ← j+1
    if v ≥ H[j]
        heap ← **true**
    else H[k] ← H[j]; k ← j
  H[k] ← v

2.2 - Theoretical Analysis

$$\sum_{i=0}^{3-1}\left[\sum_{j=0}^{n-1}\left[\sum_{k=0}^{i-1} 2\log(n-1)\right]\right] = \sum_{i=0}^{2}\left[\sum_{j=0}^{n-1} 2\log_2 * i\right] = \sum_{i=0}^{2}[n[2\log_2 * i]]$$

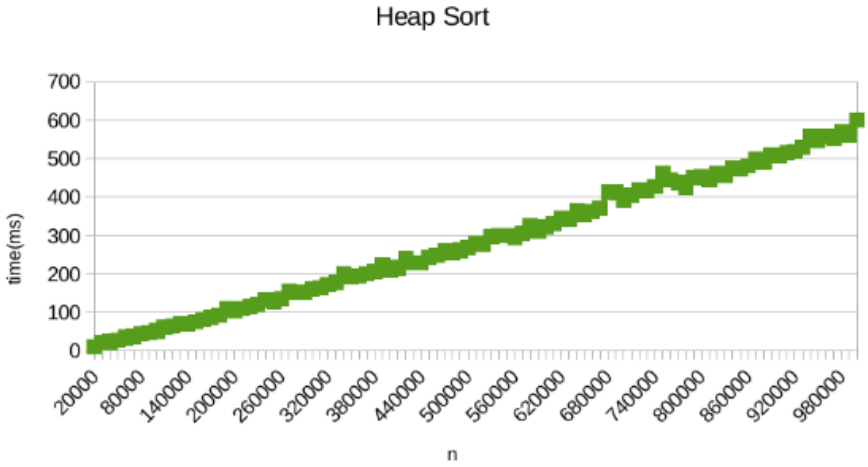
$$= [n[2\log_2 0]] + [n[2\log_2 1]] + [n[2\log_2 2]]$$

$$= [0n] + [0n] + [2n]$$

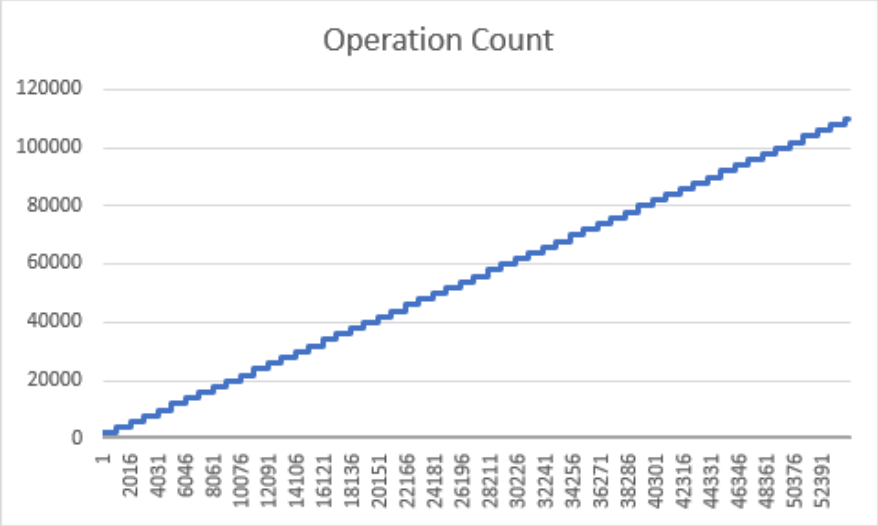
$$= 2n \in O(n\log n)$$

2.3 - Empirical Analysis

The following is the Average Time of heap sort:



The following is the Operation Count of the heap sort:



2.4 - Summary Justification of Empirical and Theoretical Analysis

The empirical results were achieved by understanding the heap sorts purpose, then deciding on the efficiency metric to be measured and the measurement unit creating an operation count vs. a time unit, furthermore, the characteristics of the input sample (range, size, etc.) was determined, and a program was then prepared to implement the heap sort algorithm. Moreover, sample inputs were generated to measure efficiency, and the program was then initialised to record the data observed. Therefore, the graphs above were generated from this program to display the heap sorts efficiency. Meanwhile, the theoretical result was achieved through characterising the size of the input which was three as the task specified for the top three most frequently borrowed tools. I then typed up a summation formula as shown above for the number of times the basic operation is performed in the worst case, and solved it using standard arithmetic rules that identified the heap sort algorithms efficiency class in big-O notation.

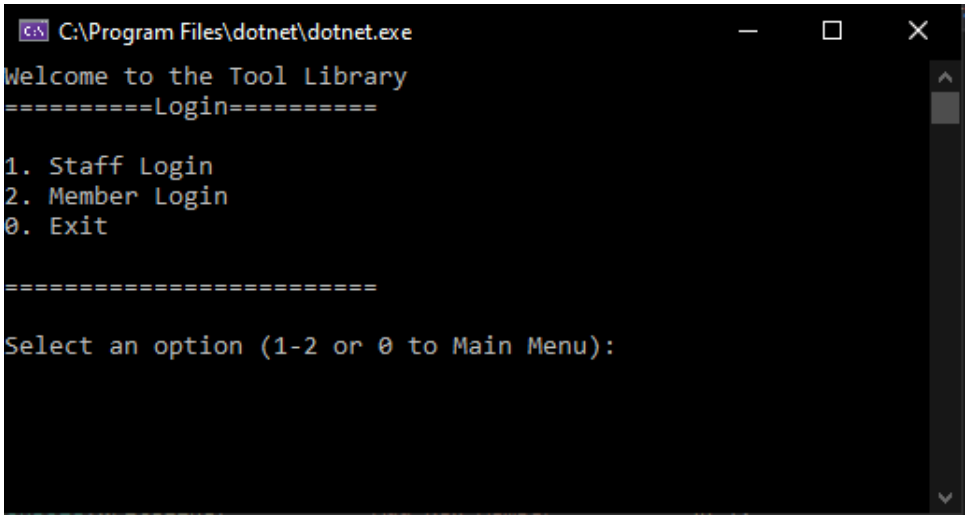
3.0 - Summary of Algorithm

After completing the empirical and theoretical analysis the heap sort is an effective sorting algorithm, but the implementation of this sort in the program was a bit difficult as it had trouble calling and temporarily storing data. Overall, the general performance of this algorithm is impressive, it's certainly one of the most efficient sorting algorithms as its worst/best/average time-complexity case is O (n log n).

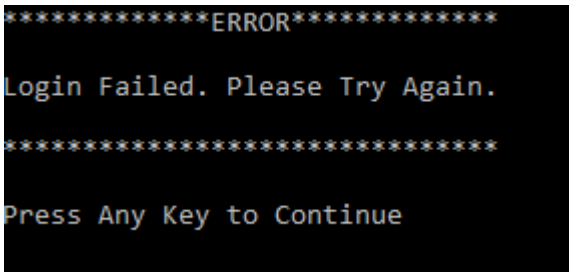
4.0 Software Test Plans and Test Results

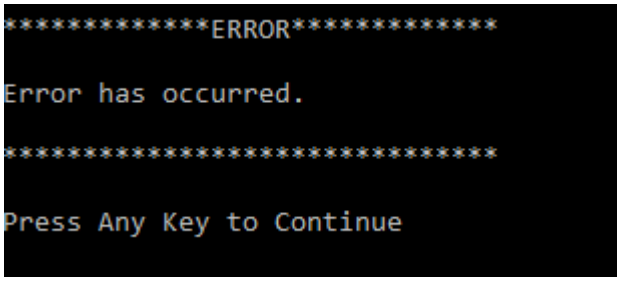
Results:

Login Screen

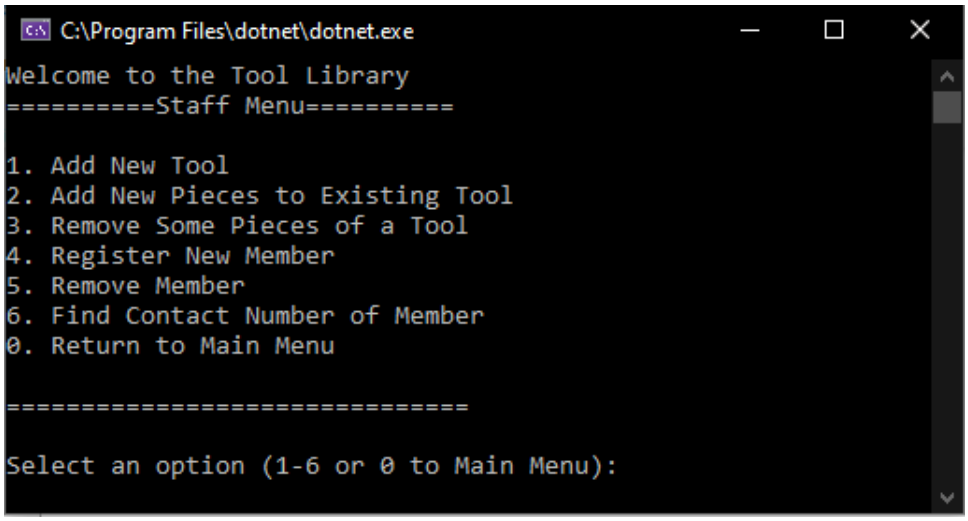


Error catching when invalid inputs a entered it throws messages similar to the one shown below

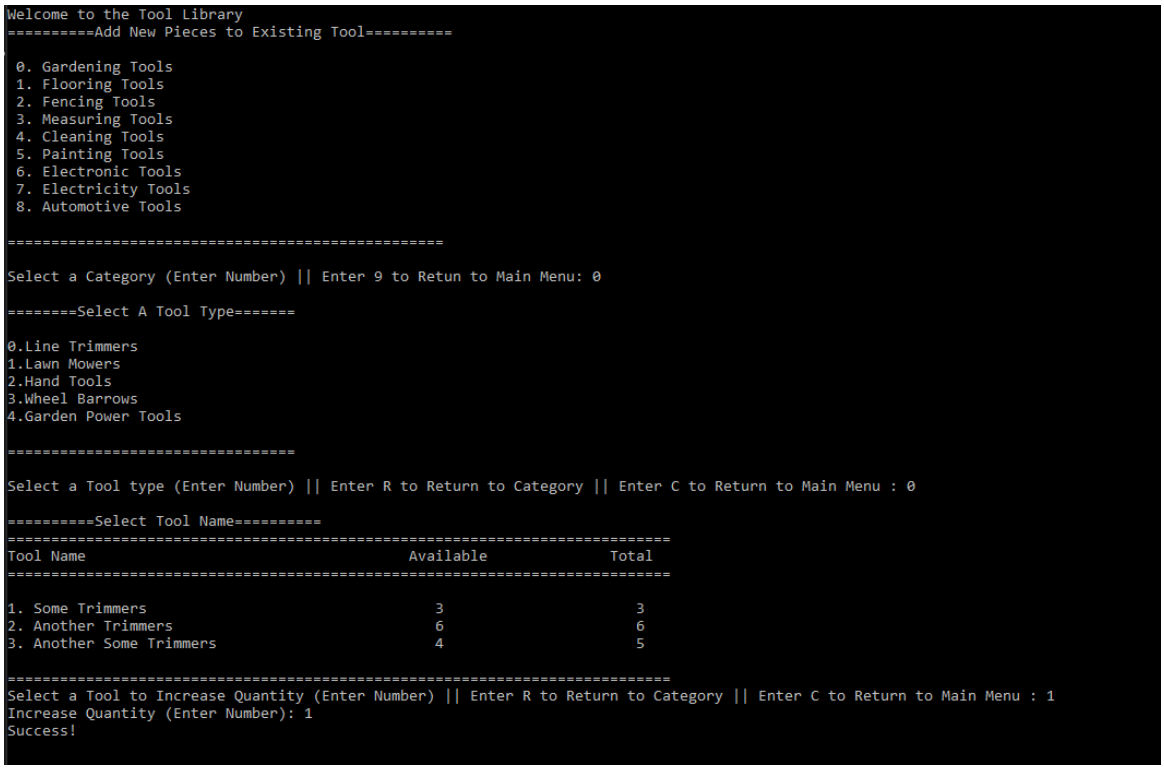




Staff Menu



Adding a New Piece to an Existing Tool



Remove Some Pieces of a Tool

```
Welcome to the Tool Library
==========Remove Some Pieces of a Tool==========

 0. Gardening Tools
 1. Flooring Tools
 2. Fencing Tools
 3. Measuring Tools
 4. Cleaning Tools
 5. Painting Tools
 6. Electronic Tools
 7. Electricity Tools
 8. Automotive Tools

================================

Select a Tool type (Enter A Number) || Enter 9 to Return to Main Menu: 0

========Select A Tool Type=======

0.Line Trimmers
1.Lawn Mowers
2.Hand Tools
3.Wheel Barrows
4.Garden Power Tools
================================

Select a Tool type (Enter Number) || Enter R to Return to Category || Enter C to Return to Main Menu : 0

==========Select Tool Name==========

================================================================================
Tool Name                                 Available             Total
================================================================================




================================================================================
Select a Tool to Remove Quantity (Enter Number) || Enter R to Return to Category || Enter C to Return to Main Menu :
```

Registering a new member

```
==========Review New Member==========

First Name: Joe
Last Name: Smith
Contact Number: 43058034985
PIN: 2222
=====================================

Enter Y to Add New Tool || Enter N to Start Over: Y
Successfully Added!
```

Remove Member (not working)

```
Welcome to the Tool Library
==========Remove Member==========

Enter Name:
```

Find Contact Number of a Member (not working)

```
==========Find Member==========

Enter First Name: John
Enter Lat Name: Doe
```

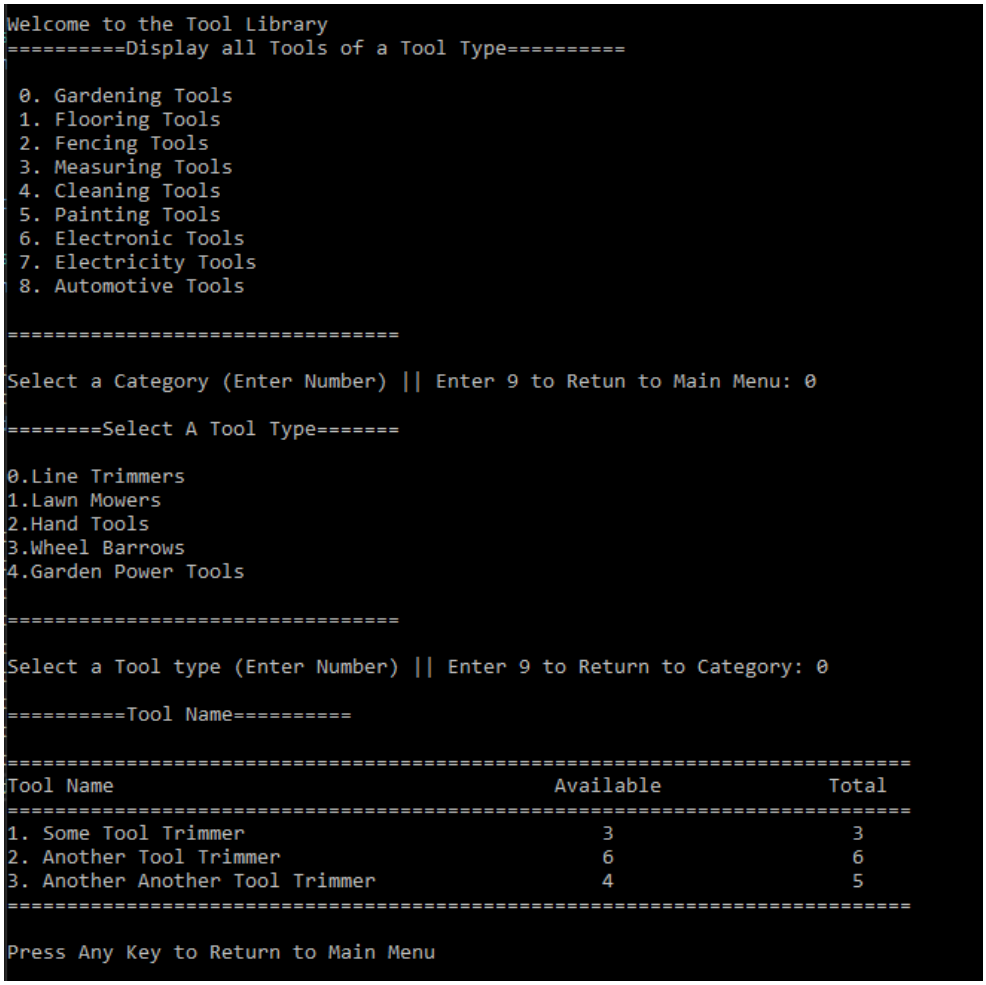
Adding a Tool

```
==========Review Added Tool==========

Tool Name: A Tool
Tool Category: Gardening Tools
Tool Type: Gardening Tools

====================================

Enter Y to Add New Tool || Enter N to Start Over: Y
Successfully Added!
```

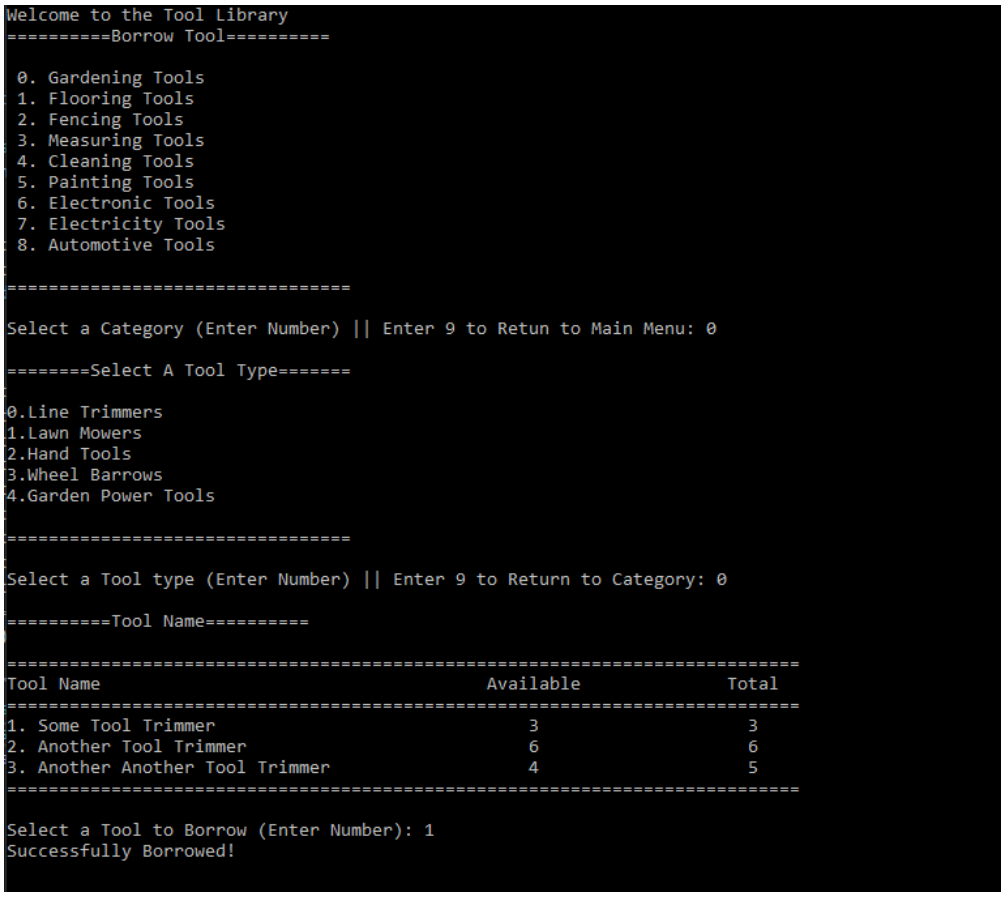
Member Menu

```
Welcome to the Tool Library
==========Member Menu==========

1. Display all Tools of a Tool Type
2. Borrow Tool
3. Return Tool
4. Currently Rented Tools
5. Top Three Most Frequentely Rented Tools
0. Return to Main Menu

==============================

Select an option (1-5 or 0 to Main Menu):
```

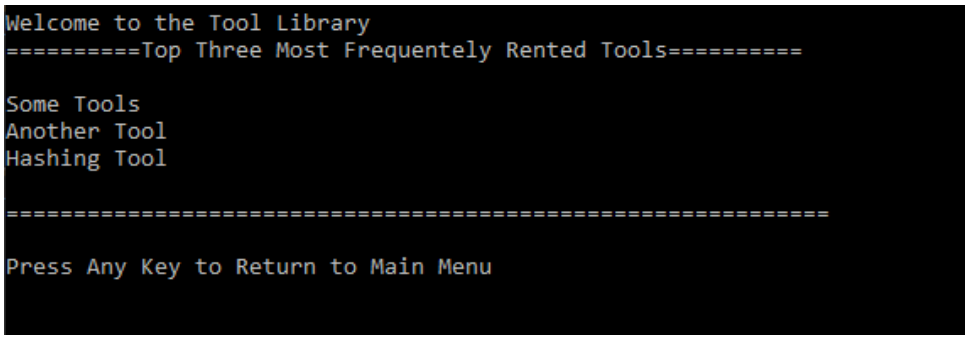
Displaying Tools by Tool Type



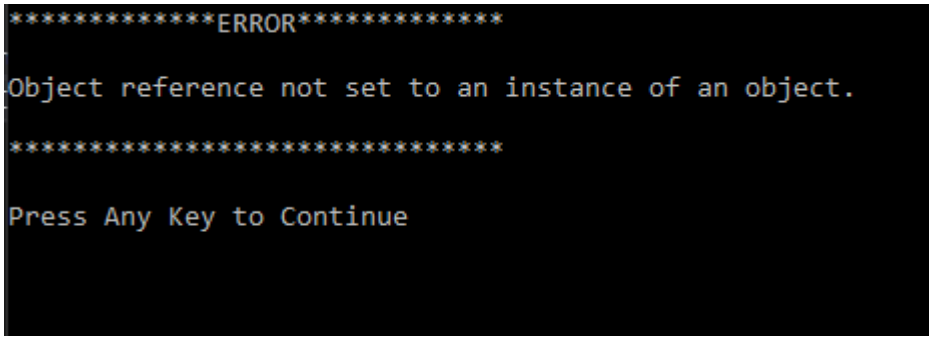
Borrowing a Tool



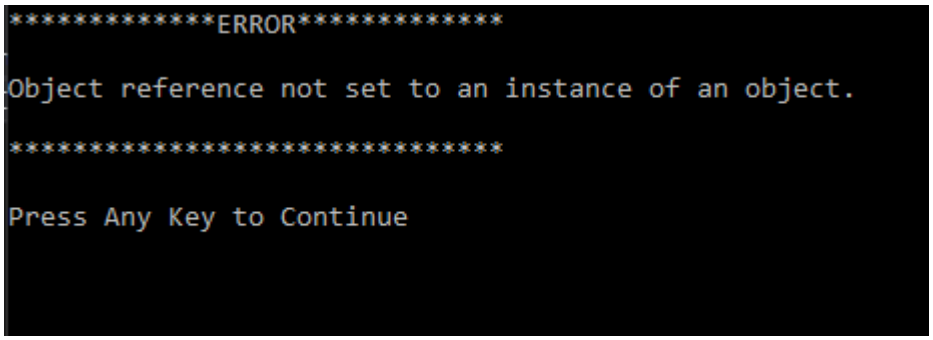
Display top 3 most frequently borrowed tools



Currently Rented Tools (not working)



Return Tool (not working)



References

Wikipedia contributors. (2021, May 11). Sorting algorithm. Wikipedia. https://en.wikipedia.org/wiki/Sorting_algorithm