ALGORITHM AND PROGRAMMING PROJECT

DIARISSO MAKAN GARBA

2019-2020

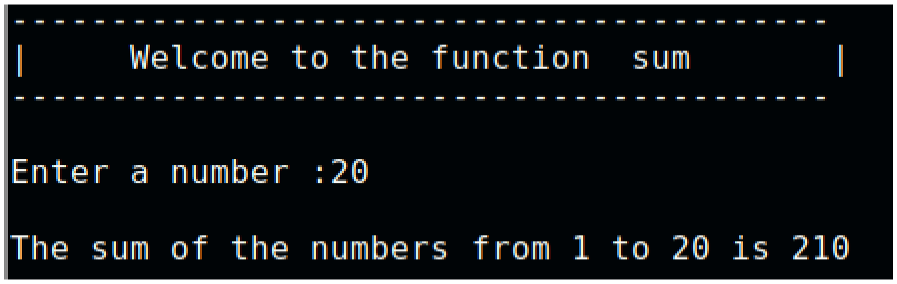
Introduction

In the WASCAL, master students are supposed to work on a project after each course. Therefore, this project focus on the Algorithm and programming course. Algorithmic is the science which study algorithms. An algorithm can be defined as a step by step operations to solve a problem. Meaning that the algorithm divides a given problem to sub-problems easy to solve. The solutions to these sub-problems lead to the general problem’s solution. In addition, a program is a software which is defined as a set of instructions written in a computer programming language in order to perform a specific task when executed. The purpose of this study is to compute a menu driven program. Thus, the program will perform the following operations: “Sum of n numbers”, “List”, “Stack”, “Queue”, “Selection Sort”, “Merge Sort”, “Sequential Search”, “Binary Search”, “Tree Traversal” .

Menu content

As declared in the introduction, the general purpose of this study is a program with some operations. So, the program has a menu which contains:

1. Sum of n numbers

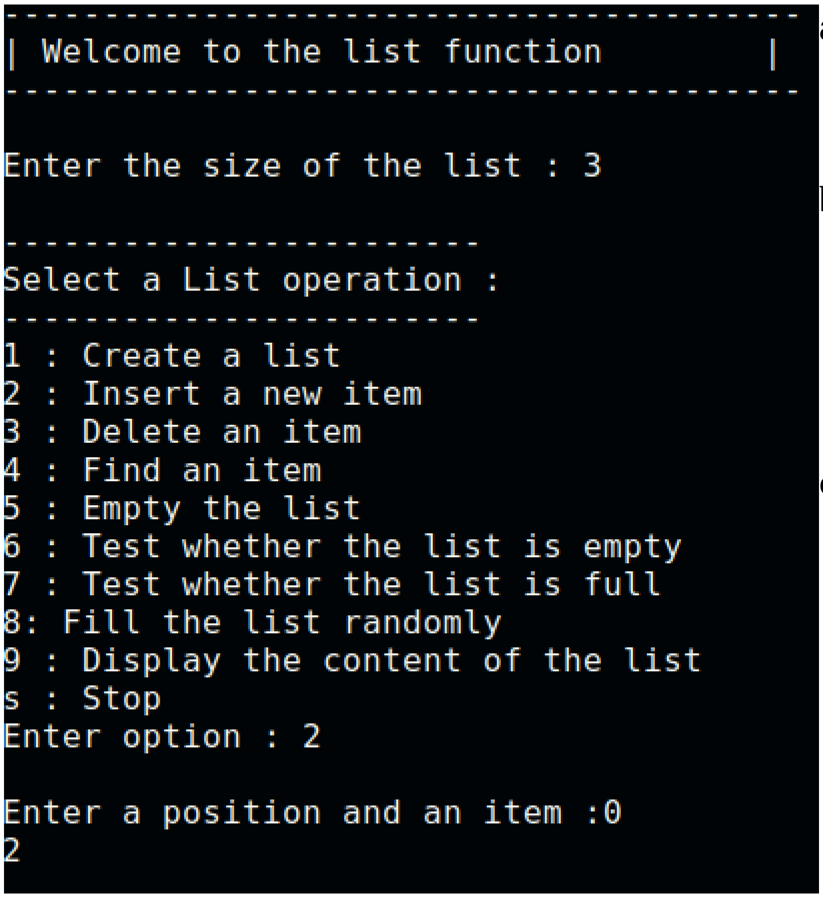


```
---------------------------------------
|     Welcome to the function  sum      |
---------------------------------------

Enter a number :20

The sum of the numbers from 1 to 20 is 210
```

This program invites the user to type a random number (integer) and compute the sum of numbers from one to the given integer. Example: When the user type 10, the program will compute 1+2+3…… +10 and return the sum which is 55. In addition, the program will guide the user when he types a negative value.

2. List

The list is a linear data type which can be filled randomly. It is possible to insert a new node at any position. Any node can also be removed from the list. One executed, the program asks the user to enter the list size before showing the menu. The operations performed in the list program are:



```
---------------------------------------
| Welcome to the list function        |
---------------------------------------

Enter the size of the list : 3

-----------------------
Select a List operation :
-----------------------
1 : Create a list
2 : Insert a new item
3 : Delete an item
4 : Find an item
5 : Empty the list
6 : Test whether the list is empty
7 : Test whether the list is full
8: Fill the list randomly
9 : Display the content of the list
s : Stop
Enter option : 2

Enter a position and an item :0
2
```

a) Create a list:
This operation creates an empty list to be filled.

b) Insert a new item:
The method used in the code is InsertNode. The user will specify the position at which the item will be inserted before typing the item itself.

c) Delete an item:
The performing method of this operation is DeleteNode. In this operation, just giving the item's position will make the user delete the item.

d) Find an item :
Once typed, an item is found at a specific position. The FindNode method performs this operation.

e) Empty the List :
Empty the list when the option is selected. The method MakeEmpty is performing this operation.

f) Test whether the list is empty or not :
Performed by the IsEmpty method, this operation test if the list contains items or not.

g) Test whether the list is full :
Performed by the IsFull method, this operation test if a new item can be added or not.

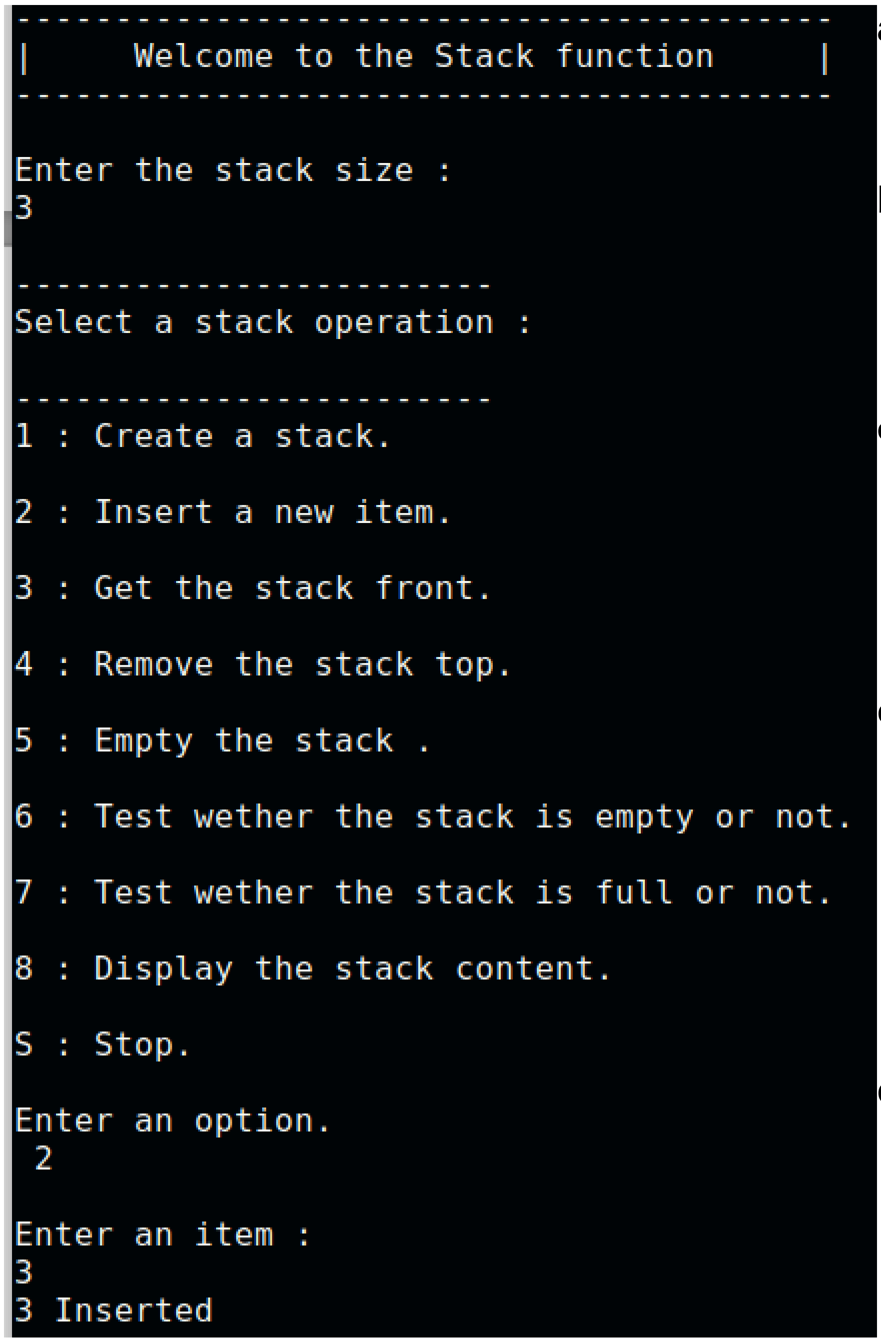
h) Fill the list randomly :
The RandomNode method performs this operation. It fills the list with random values.

i) Display the content of the list :
Display the list size, the number of elements in the list and all the items and their indexes that the list contains. It is performed by the Display method which use the Size and NbElements methods.

j) Stop :
Quit the list operation and return the user to the general menu. It is the default option.

3. Stack

The stack is also a linear data type. In contrast with the list, the stack items are added and removed only by the top. This is called the LIFO(Last In First Out) processus. The stack program provide the operations like :



```
-----------------------------------------
|     Welcome to the Stack function     |
-----------------------------------------

Enter the stack size :
3

------------------------
Select a stack operation :

------------------------
1 : Create a stack.

2 : Insert a new item.

3 : Get the stack front.

4 : Remove the stack top.

5 : Empty the stack .

6 : Test wether the stack is empty or not.

7 : Test wether the stack is full or not.

8 : Display the stack content.

S : Stop.

Enter an option.
 2

Enter an item :
3
3 Inserted
```

a) Create a stack :

This operation creates an empty stack to be filled.

b) Insert a new item :

The method used in the code is Push. The user will specify the item to be inserted.

c) Get the stack top :

The method used to implement it is Peek. This will make the user display the top item of the stack

d) Remove the stack top :

The performing method of this operation is Pop. In this operation, just selecting the option will make the user remove the top item.

e) Empty the stack :

Empty the stack when the option is selected. The method MakeEmpty is performing this operation.

f) Test whether the stack is empty or not :

Performed by the IsEmpty method, this operation test if the stack contains items or not.

g) Test whether the stack is full or not:

Performed by the IsFull method, this operation test if a new item can be added or not.

h) Display the stack content :

Display the stack size, the number of elements in the stack and all the items and their indexes that the stack contains. It is performed by the Display method which use the Size and NbElements methods.

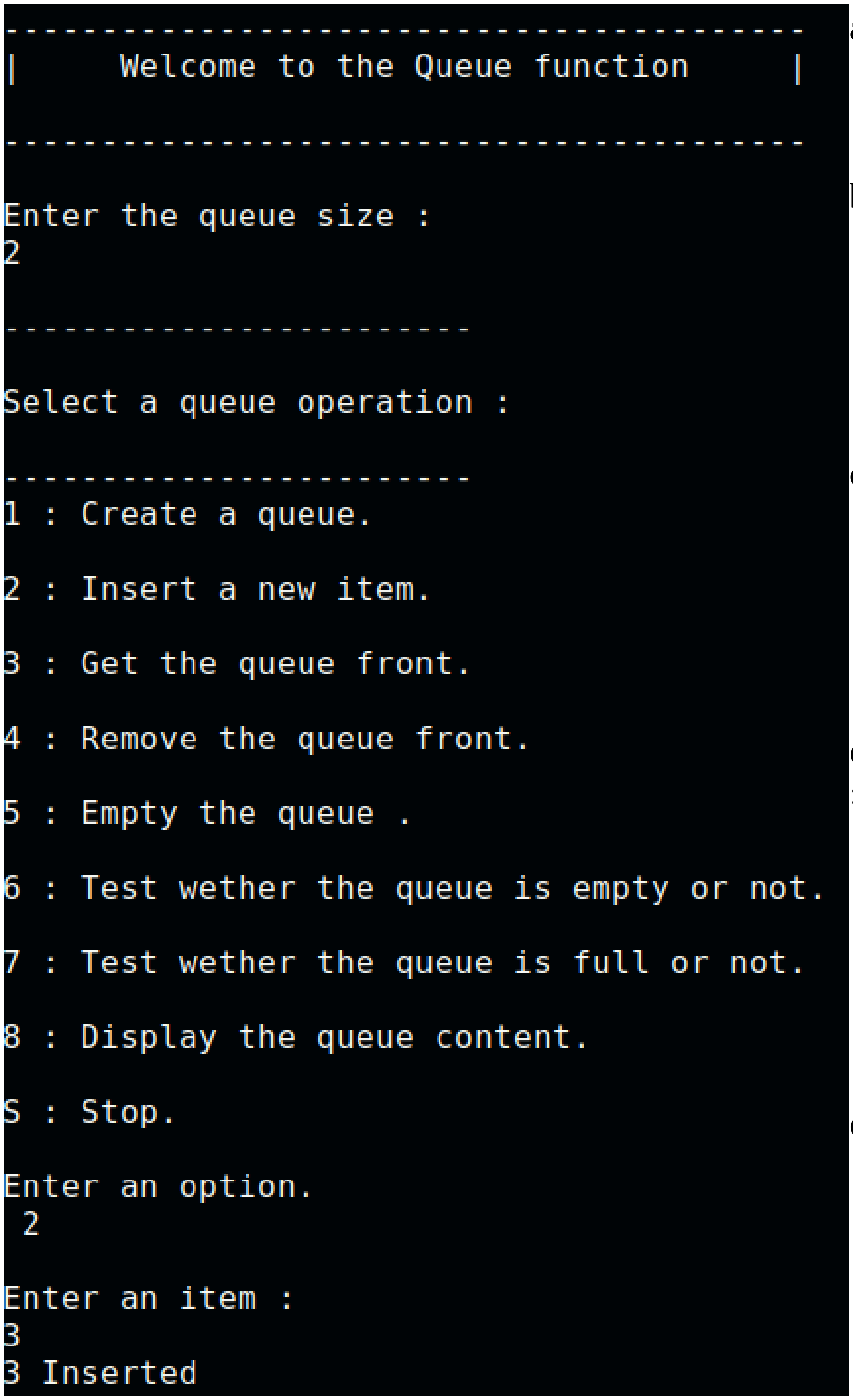
i) Stop :

Quit the stack operation and return the user to the general menu. It is the default option.

NB: All the queue operations have the same time complexity O(1).

4. Queue

This is another linear data type but different from the list by some of its peculiarity. In contrast with both list and stack, an item can be added only at the back (rear) in a queue and also the removing is done by the front of the queue. That is called the FIFO(First In First Out) processus. The queue program contains the following operations :



```
----------------------------------------
|      Welcome to the Queue function     |

----------------------------------------

Enter the queue size :
2

------------------------

Select a queue operation :

------------------------
1 : Create a queue.

2 : Insert a new item.

3 : Get the queue front.

4 : Remove the queue front.

5 : Empty the queue .

6 : Test wether the queue is empty or not.

7 : Test wether the queue is full or not.

8 : Display the queue content.

S : Stop.

Enter an option.
 2

Enter an item :
3
3 Inserted
```

a) Create a queue :

This operation creates an empty queue to be filled.

b) Insert a new item :

The method used in the code is Enqueue. The user will specify the item to be inserted.

c) Get the queue front :

The method used to implement it is Peek. This will make the user display the top item of the stack

d) Remove the queue front :

The performing method of this operation is Dequeue. In this operation, just selecting the option will make the user delete the item.

e) Empty the queue :

Empty the queue when the option is selected. The method MakeEmpty is performing this operation.

Test whether the queue is empty or not : performed by the IsEmpty method, this operation test if the queue contains items or not.

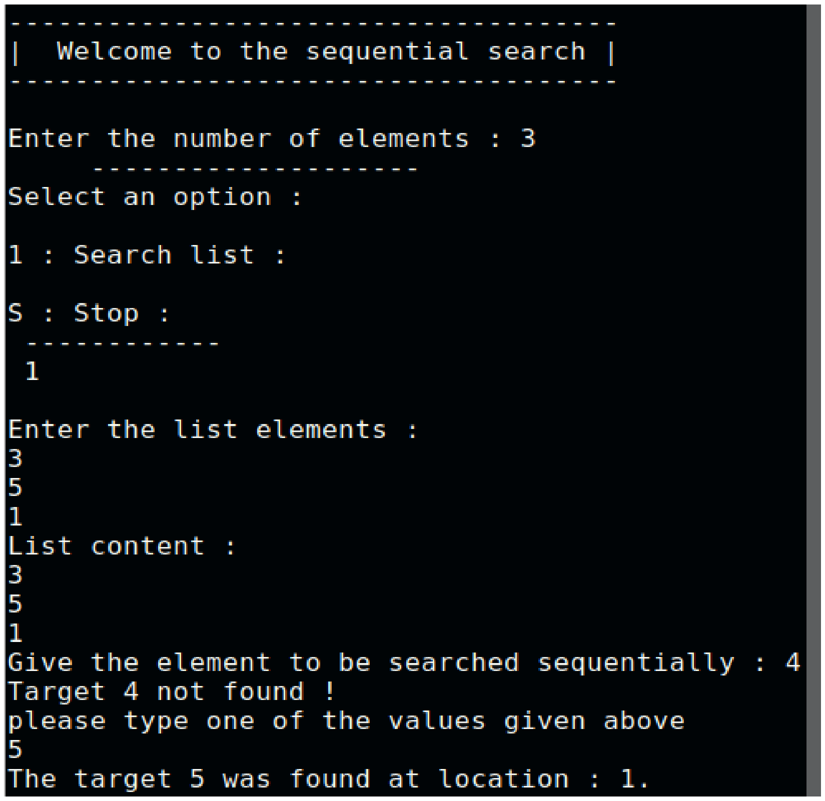
f) Test whether the list is full :
Performed by the IsFull method, this operation test if a new item can be added or not.

g) Display the queue content :
Display the queue size, the number of elements in the queue and all the items and their indexes that the queue contains. It is performed by the Display method which use the Size and NbElements methods.

h) Stop :
Quit the queue operation and return the user to the general menu. It is the default option.

i) NB :
As the stack, all the queue operations have the same time complexity O(1).

5. Sequential search

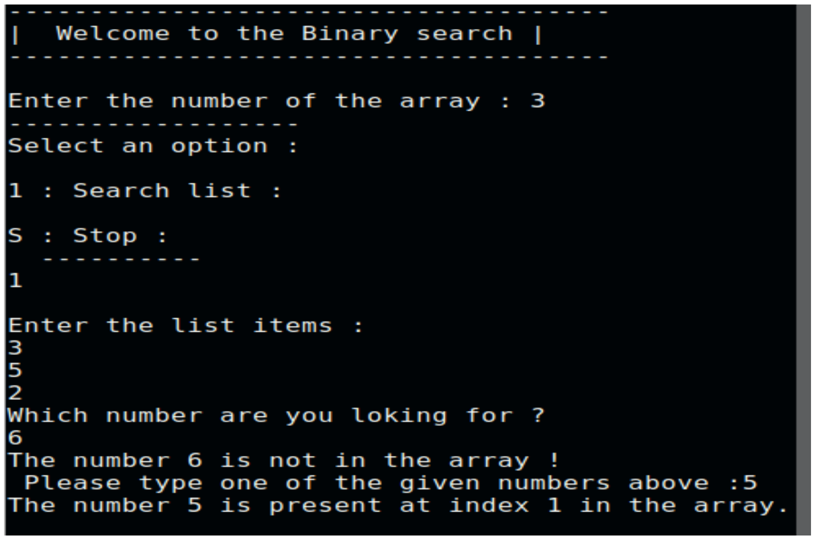


```
-------------------------------------
|  Welcome to the sequential search |
-------------------------------------

Enter the number of elements : 3
     --------------------
Select an option :

1 : Search list :

S : Stop :
 ------------
 1

Enter the list elements :
3
5
1
List content :
3
5
1
Give the element to be searched sequentially : 4
Target 4 not found !
please type one of the values given above
5
The target 5 was found at location : 1.
```

The purpose of this program is to f ind a given item. The user is pleased to type the number of elements of the list at the beginning. A menu appears and let him select an option. The “Stop” option ends the function and return the user to the home menu. The “search list” option let the user continues with the list operations. Then the list elements are typed one by one till the given size is reached. The computer displays the list content and ask the user to type the item to be searched sequentially. Moreover, when the given item is not in the list, the user is guided. Finally**,** the searched item and its position will be displayed. The list is created using an array.

6. Binary search

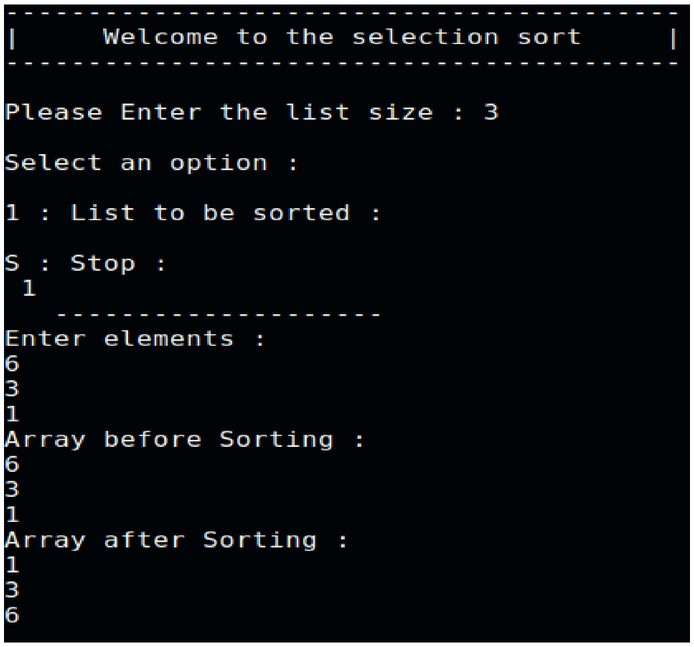
As the sequential search the binary search program purpose is also to find a given item in a sorted array. The process starts with the whole array. Then it is repeatedly halved until the required item is obtained or the remaining array is empty. If the required item is greater than the middle item, then the upper half of the array is considered. Otherwise, the lower half is considered.



```
-------------------------------------
|  Welcome to the Binary search |
-------------------------------------

Enter the number of the array : 3
------------------
Select an option :

1 : Search list :

S : Stop :
  ----------
1

Enter the list items :
3
5
2
Which number are you loking for ?
6
The number 6 is not in the array !
 Please type one of the given numbers above :5
The number 5 is present at index 1 in the array.
```

The program will let the user type the number of elements of the list at the beginning. A menu appears and let him select an option. The “Stop” option ends the function and return the user to the home menu. The “search list” option let the user continues with the list operations. Then, the list elements are typed one by one till the given size is reached. The computer displays the list content and asks the user to type the item to be searched. Moreover, when the given item is not in the list, the user is guided. Finally, the searched item and its position will be displayed. The list is created here using an array also.

7. Selection sort

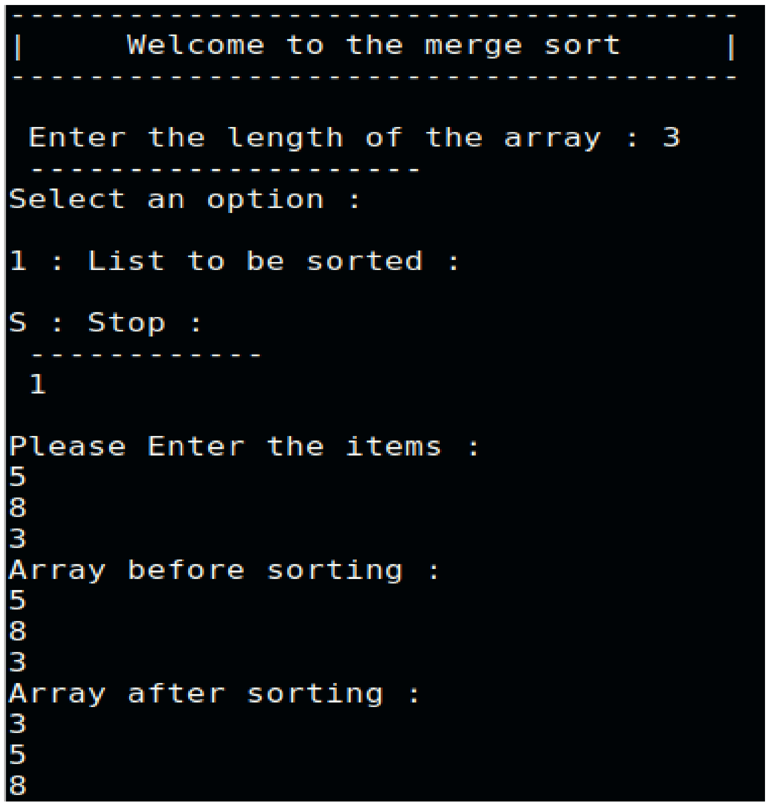


```
-----------------------------------------
|     Welcome to the selection sort     |
-----------------------------------------

Please Enter the list size : 3

Select an option :

1 : List to be sorted :

S : Stop :
 1
   --------------------
Enter elements :
6
3
1
Array before Sorting :
6
3
1
Array after Sorting :
1
3
6
```

This program sorts a list of values in ascendant order. The list is divided into two parts. All the elements are sorted in one part and the other part is not sorted. The minimum element of the array is placed at the beginning by replacing with the beginning element. At the end of the process, the list is sorted.

The user is pleased to type the size of the array at the beginning. A menu appears and let him select an option. The “Stop” option ends the function and return the user to the home menu. The “List to be sorted” option let the user continues with the list operations. Then, the list elements should be typed one after another till the array size is reached. The program will return the unsorted and sorted list. The time complexity of the selection sort is O(n^2).

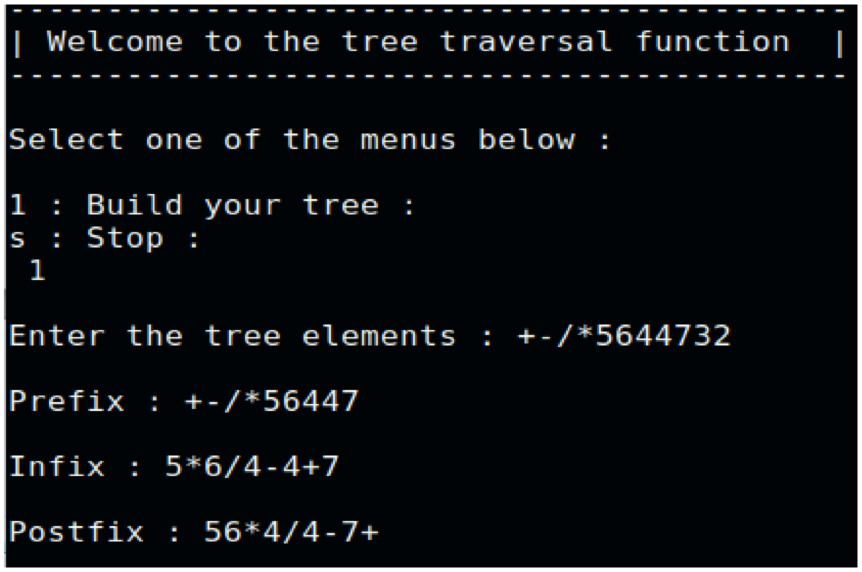
8. Merge sort



```
-------------------------------------
|     Welcome to the merge sort     |
-------------------------------------

 Enter the length of the array : 3
 --------------------
Select an option :

1 : List to be sorted :

S : Stop :
 ------------
 1

Please Enter the items :
5
8
3
Array before sorting :
5
8
3
Array after sorting :
3
5
8
```

This program also sorts a list of values in ascendant order. The technique used is the divide and conquer one. The user is pleased to type the size of the array at the beginning. A menu appears and let him select an option. The “Stop” option ends the function and return the user to the home menu. The “List to be sorted” option let the user continues with the list operations Then, the list elements should be typed one after another till the array size is reached. The program will return the unsorted and sorted list. In this specific program the list is divided into two parts at each searching stage, and then the left and right sub-arrays are merged to give one sorted array. The time complexity of merge sort is O(n).

9. Tree Traversal(prefix, infix, postfix):



```
-------------------------------------------
| Welcome to the tree traversal function  |
-------------------------------------------

Select one of the menus below :

1 : Build your tree :
s : Stop :
 1

Enter the tree elements : +-/*5644732

Prefix : +-/*56447

Infix : 5*6/4-4+7

Postfix : 56*4/4-7+
```

The tree is a non linear data structure in which each node has at most two children. The tree prefix is to visit respectively the root, the left sub-tree and finish with the right sub-tree. While the tree infix is to visit the left sub-tree, the root and finish with the right sub-tree, the postfix starts visiting the left, right sub-trees and ends with the root.

Suggestions :

As each programmer, the beginning was very difficult, because while the written program were performing most of the menu operations, all of them were not working as perfect as they could be. That is sometimes due to the difficulty to understand some programming aspects before moving forward. So, the main problems encountered were somehow lot. One of them was that the list code contained some errors. Another was that the sorting algorithms seen in class like the selection sort were not matching as well. The swapping function should be swap(Array[min], Array[i]) instead of swap(Array[i], Array[min]). The biggest issue I have had was that I didn’t understand well the course. Perhaps the course level was higher for me, but I am reading more about it. Moreover, we did not had lot of classes in coding that is what explains the difficulties to understand some coding aspects. To sum up, I am performing my knowledge in this field and most of my written codes are working well.

Summary:

This work studied how to compute some algorithms using the programming language C++. All the functions (“Sum of n numbers”, “List”, “Stack”, “Queue”, “Selection Sort”, “Merge Sort”, “Sequential Search”, “Binary Search”, “Tree Traversal”) were developed in the same program using a class and a void method for each function. The single main method were used for all the code source. There is where the functions are called by using a character value implemented in a while loop to let the user make a selection in the menu, and a switch functions to implement all the case in the menu.

References:

http://www.softwareandfinance.com/

https://www.geeksforgeeks.org/

https://www.tutorialspoint.com/index.htm

https://www.techiedelight.com/

https://www.programiz.com/dsa/

Algorithm and Programming (Prof. Harouna NAROUA)

Table of Contents