Group Assignment
Data Structures and Algorithms
IS2110

Group - 20

17020212 - dasith84@gmail.com
17020565 - duminduchamal@gmail.com
17020883 - thushanthywalter10@gmail.com

Question 01 - This is done by D. N. Dewapriya(17020212)

The implementation is done using C++ language, so can be compiled and run by any IDE which supports C++(eg:-DEV C++).

Implementation is done using 4 functions;

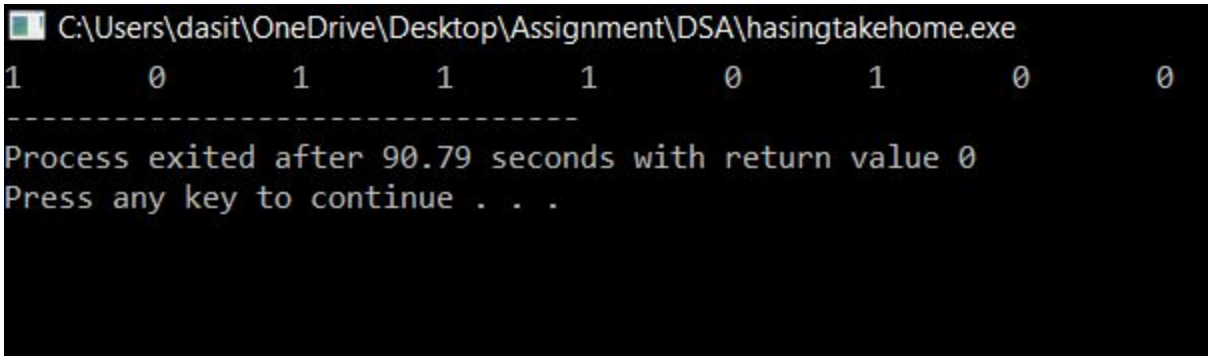
1. main function
2. make_array function
3. hashtable function
4. get_sum function

Make_array function is used to get the data from the Hashint.txt file to an array to later implement into a hash table. The file is read as a stream and inserted into an array line by line.

Hashtable function is used to create the hash table, I have used
Linear probing to implement the table. 100003 was the nearest prime number so the data was modulated by that number to get the index. Which was then incremented using a while loop to solve collisions.

Get_sum function was used to see whether there are pairs which can add up to get the target sum. In this function variable x is allocated with the target sums using a for loop after subtracting the target sum with the hashtable values the answer is hashed and checked whether that number is in the hash table; if no, the hashtable is probed to see whether that number exists in the hashtable if so 1 is printed if not 0 is printed.

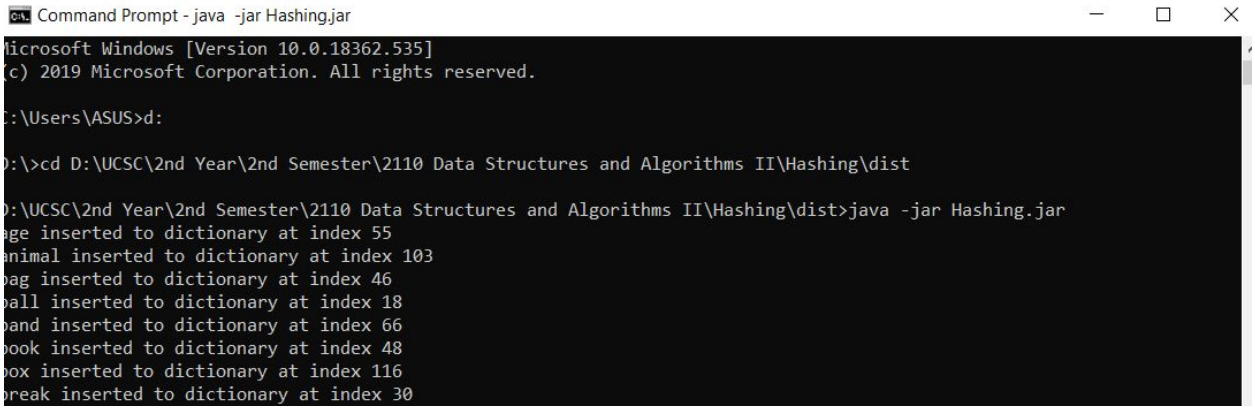
It takes about 1 1/2 minutes to complete the programme.



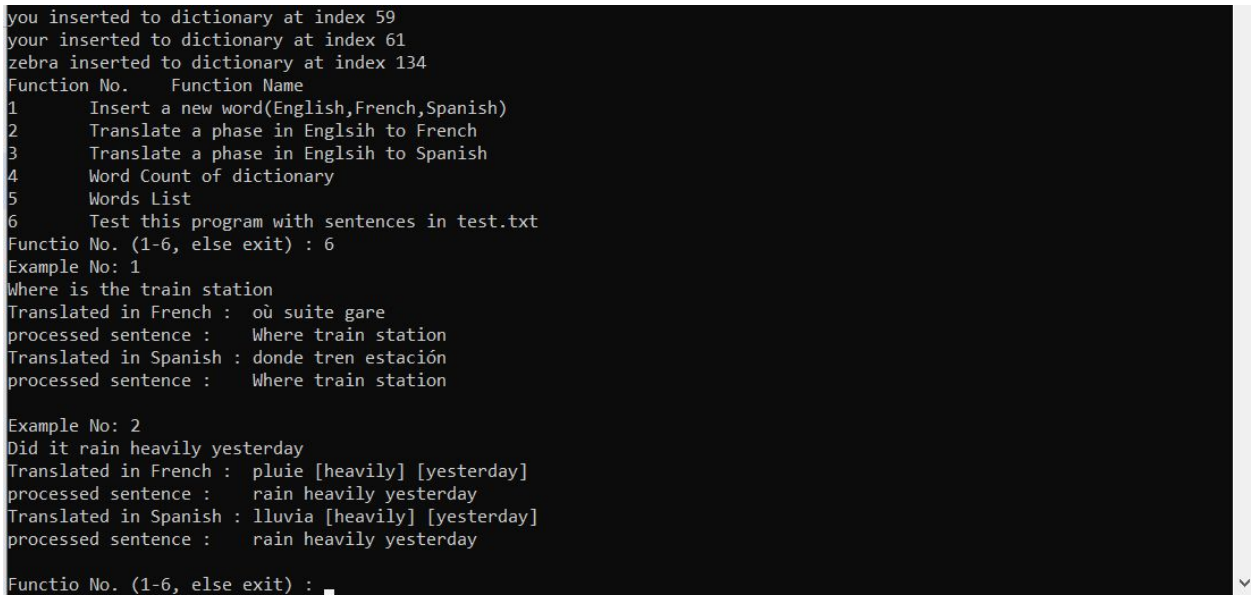
-output-

Question 02 - This is done by D.C.H. Muthukumarage(17020565)

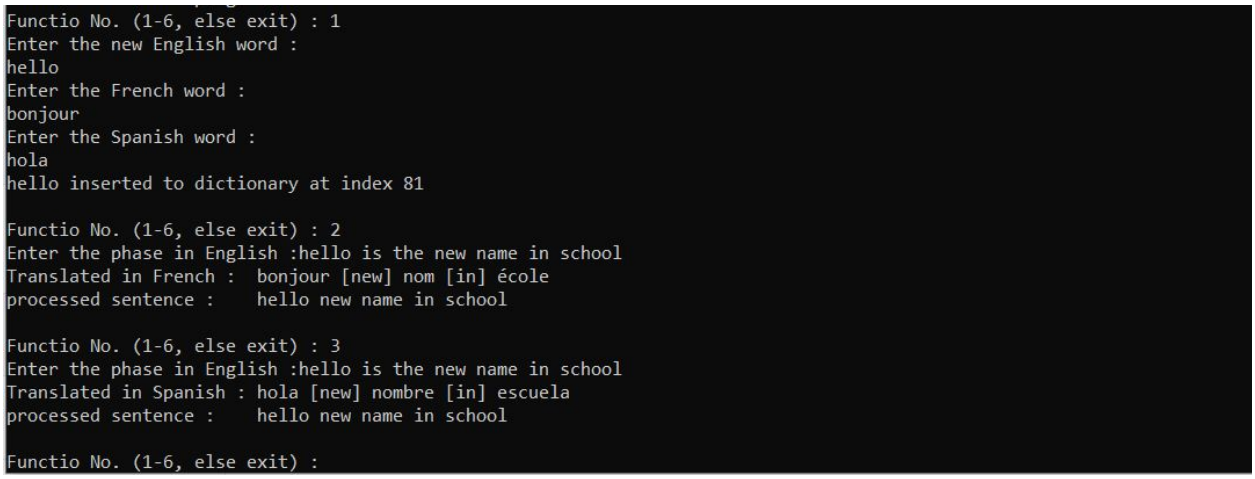
To execute the program please go inside the hashing folder and go inside the dist folder. Then you will find the “Hashing” executable jar file. Then come to this location in command prompt and type “java -jar Hashing.jar”. Then program will execute.



In the same location I have created a text.txt file. There are sample sentences in English. To translate it into French and Spanish please enter 6.



As you can see we can insert a new word to the dictionary. As well as that we can translate an English phrase into French phrase and English phrase into Spanish phrase.



In the above screenshot I have entered “hello” word into the Dictionary. And I have given the functionality of getting the word count in the dictionary and see the words list. Implementation has been done by JAVA with the help of NetBeans IDE. In the code I have created a class with string 2D array.

Hash function :

In the hash function firstly I have converted the string to lowercase. Since ascii values differ from uppercase to lowercase. Here I have taken the lowercase ascii value and multiply it with base 33 as in the polynomial accumulation and add all those ascii values of the string. I have used polynomial accumulation because palindromes have the same summation of ascii values without considering the position. Then I have mode it with table size. Then the result is taken as the array index.

Insert function :

In the insert function I have sent the English string to the hash function and took the hash value as the Index and insert the English word, French word and Spanish word in the 2D array. For collisions I have followed linear probing.

Search function :

In the search function firstly I have splitted English phrase in to words by space. Then I have kept a word list in the ignore.txt file which contains words which have to be omitted when translating. Then every word in the English phase is compared with that ignored word list and if it is the same loop will increment and get the next word. If not it will hash the value and see there is that English word in the hashtable. If it is there it will print the corresponding French word or Spanish word as user asked to translate into. If english word is not in the hashtable then it will print the same english word in between “[ ]”.

Question 03 - This is done by W. I. Thushanthy (17020883)

The implementation is done using C++ language.To execute the program run the question03.cpp file.

Functions

- forward_Hash()
- backward_Hash()
- isPalindrome()
- findInteger()
- main()

forward_Hack()

Filing the forward hash table with calculated hash value in the same index of the character. When going through this process the value stored with sum of the previous characters hash value + current character hash value In the order of 0th index to Nth index.

backward_Hash()

Filing the backward hash table with calculated hash value in the same index of the character. When going through this process the value stored with sum of the previous characters hash value + current character hash value In the order of Nth index to 0th index(backward direction).

isPalindrome()

This function will return a boolean value whether the sub string is a palindrome or not. To check the function accepts the starting and last indexes of the substring and comparing the hash value difference between forward and backward hash value of the indexes. Ex :

if((forward[last] - forward[start]) == (backward[start] - backward[last]))

findInteger()

This function is used to find the starting index and last index of the query from user input.(Incase the user entered the query with spaces.)

main()

This is driver function for the program, which we used to input the arguments and Inspecting the Output.

#Execution process

Run the program…

```
Enter the String : -step on no pets
```

Enter the string you want to test.

```
Enter the String : -step on no pets
Enter the Number of Queries : 4
```

Enter the number of Queries going to be generated.

```
Enter the String : -step on no pets
Enter the Number of Queries : 4
Enter query [ starting_index , ending_index ] :[0,14]
Enter query [ starting_index , ending_index ] :[0,3]
Enter query [ starting_index , ending_index ] :[5,9]
Enter query [ starting_index , ending_index ] :[10,14]
```

Enter the queries one by one with the format given front.

```
Enter the String : -step on no pets
Enter the Number of Queries : 4
Enter query [ starting_index , ending_index ] :[0,14]
Enter query [ starting_index , ending_index ] :[0,3]
Enter query [ starting_index , ending_index ] :[5,9]
Enter query [ starting_index , ending_index ] :[10,14]
[0,14] - is a Palindrome
[0,3] - is a Palindrome
[5,9] - is a Palindrome
[10,14] - is not a Palindrome
```

Finally you see the output which will give the result, whether the sub string is a palindrome or not.