IS 2110

Data Structures and Algorithms II

Group-04

Group Members

Team member	Contribution	Email Addresses	Index number
AHILAN A.	Question 1	archanaahil@gmail.com	17020042
JAYATHILAKA H.A.P.C.J.	Question 2	mailtochamodij@gmail.com	17020387
RATHNAYAKE A.R.M.M.K	Question 3	kavindirathnayake0@gmail.com	17020727

Question 1

Implement the hash table-based algorithm and determine, for each of the 9 target sums x, whether or not x can be formed as the sum of two entries in the given array.

How to solve it ?

- Developed this algorithm using c++.
- There are two functions,

  1.main function

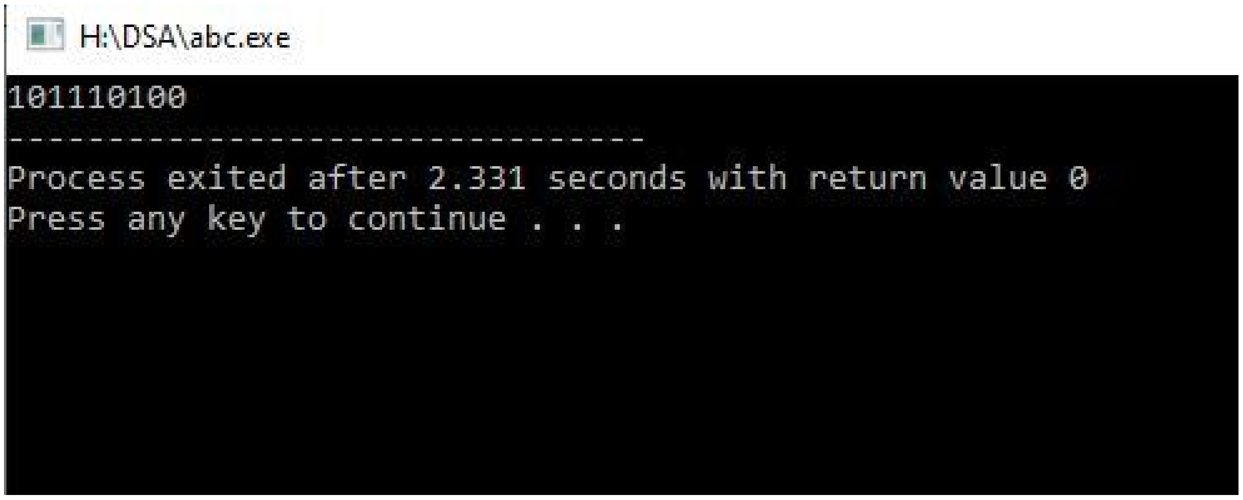
  2.findSum function

- findSum function will create the hashmap and insert the elements into the hashmap. Then it will findout the complement.

  complement= targetsum - selected value.

- Using a for loop it will check that whether is there any value in the hashmap which is equal to complement. If that true function return, string type output 1 Else 0.
- In the main function I used "ifstream inp("HashInt.txt");" to read the HashInt.txt file and also insert those sum value given into the array.
- By call the findSum function we can get the output.
- Parameters for findSum function are (array, size of that array,targetsum).

Output:



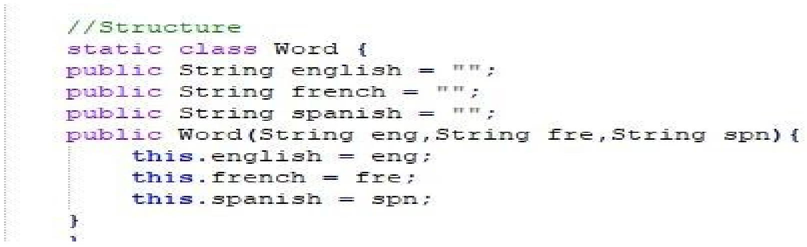
```
H:\DSA\abc.exe
101110100
--------------------------------
Process exited after 2.331 seconds with return value 0
Press any key to continue . . .
```

Question 2

<u>Translate a sentence from one language to another using a dictionary and word-for-word substitution.</u>

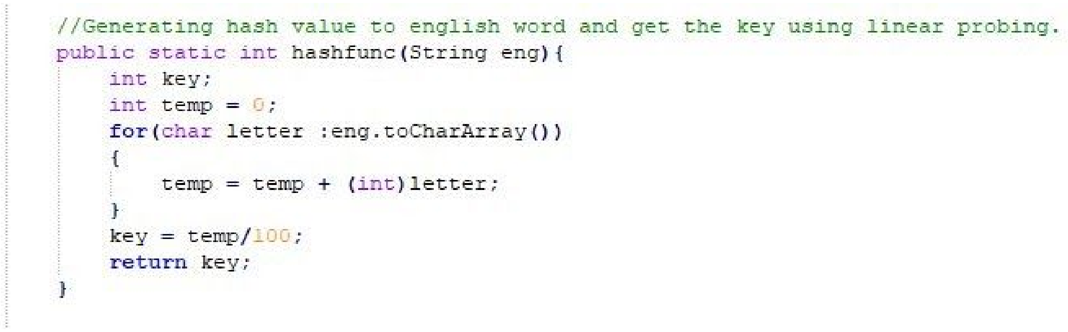
How to solve it ?

- Firstly we have to convert the given sample dictionary (dictionary.xlx) to the CSV file and create another CSV file which contains the English sentences. Here we used two input files as before mentioned.
- We implement this code using Java.
- We should create an ArrayList to store the English, French and Spanish word.



```
//Structure
static class Word {
public String english = "";
public String french = "";
public String spanish = "";
public Word(String eng,String fre,String spn){
    this.english = eng;
    this.french = fre;
    this.spanish = spn;
}
```

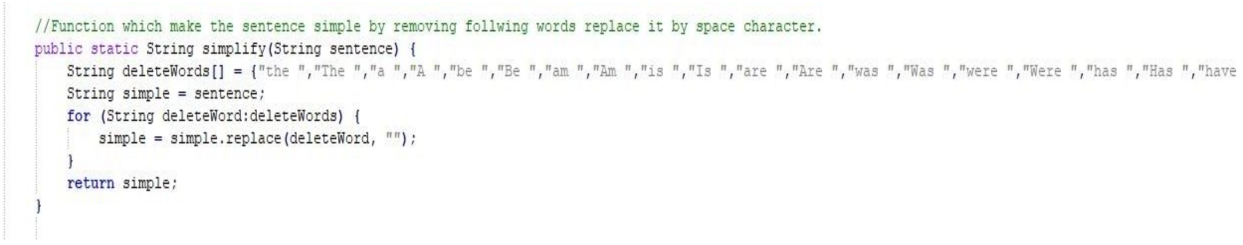
- Next, we take the dictionary word file to a variable and split symbol as "," to another variable. Call both these two variables in the dictionaryLoder function.
- In there the line by line the CSV file gets read and split the line into words by "," symbol and put it into a string temp array.
- From the zeroth index of the temp array, the key is generated. It is generated by using a hash function.



```
//Generating hash value to english word and get the key using linear probing.
public static int hashfunc(String eng){
    int key;
    int temp = 0;
    for(char letter :eng.toCharArray())
    {
        temp = temp + (int)letter;
    }
    key = temp/100;
    return key;
}
```

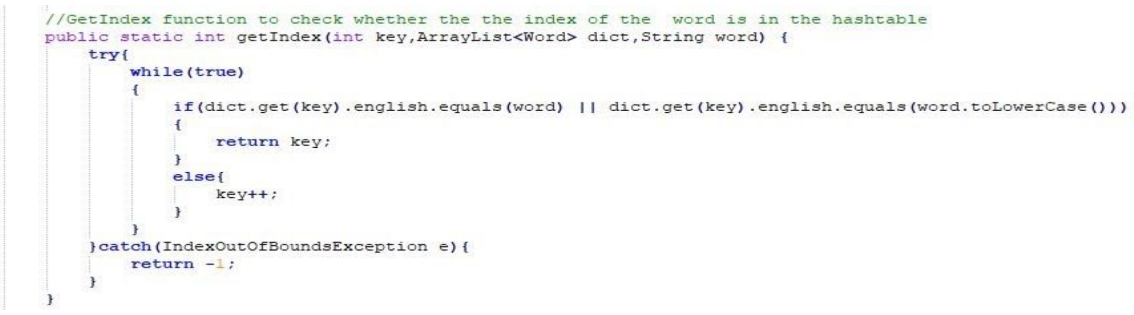
- Using the finder function we find out the position in the hashtable.
- After that, we should read the next CSV file and split their words using space characters.
- Firstly we should print the original sentence. After that to get the simple process sentence

, by using the simplify function we remove the hardcoded words and replace it using an empty string.



```
//Function which make the sentence simple by removing follwing words replace it by space character.
public static String simplify(String sentence) {
    String deleteWords[] = {"the ","The ","a ","A ","be ","Be ","am ","Am ","is ","Is ","are ","Are ","was ","Was ","were ","Were ","has ","Has ","have
    String simple = sentence;
    for (String deleteWord:deleteWords) {
        simple = simple.replace(deleteWord, "");
    }
    return simple;
}
```

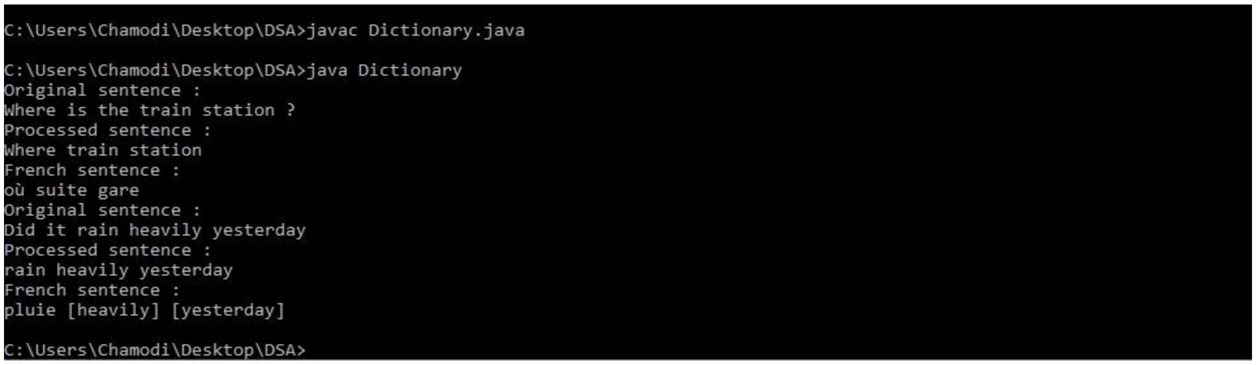
- Get the process sentence and again split using space character. Get the words individually from it and get the hash value to that word using the above hash function.
- In most of the case, the first later is a capital letter in the word but in some instance, first letter will be simple. So, firstly we check the capital letter started word and if not found we move on to it simple case letters.



```
//GetIndex function to check whether the the index of the  word is in the hashtable
public static int getIndex(int key,ArrayList<Word> dict,String word) {
    try{
        while(true)
        {
            if(dict.get(key).english.equals(word) || dict.get(key).english.equals(word.toLowerCase()))
            {
                return key;
            }
            else{
                key++;
            }
        }
    }catch(IndexOutOfBoundsException e){
        return -1;
    }
}
```

- From that, we found the relevant index to that word. If found, we get the relevant french word and if not found write the English word between the square brackets.

Output:



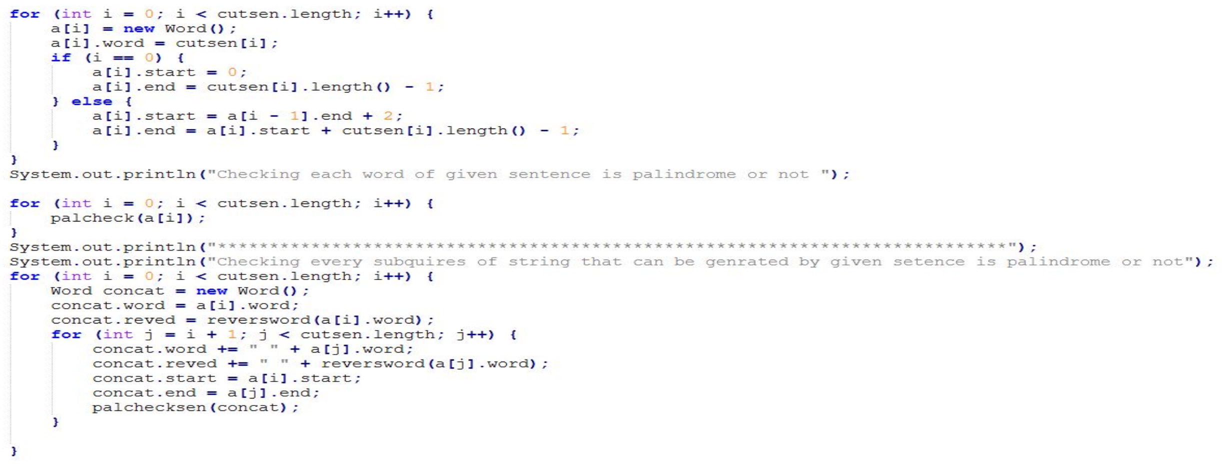
```
C:\Users\Chamodi\Desktop\DSA>javac Dictionary.java

C:\Users\Chamodi\Desktop\DSA>java Dictionary
Original sentence :
Where is the train station ?
Processed sentence :
Where train station
French sentence :
où suite gare
Original sentence :
Did it rain heavily yesterday
Processed sentence :
rain heavily yesterday
French sentence :
pluie [heavily] [yesterday]

C:\Users\Chamodi\Desktop\DSA>
```

Question 3

<u> Consider that substrings of a given input string are queried in terms of the indexes of the substrings to check whether the substrings are palindromes or not. For example, madam, noon and refer are palindromes.</u>

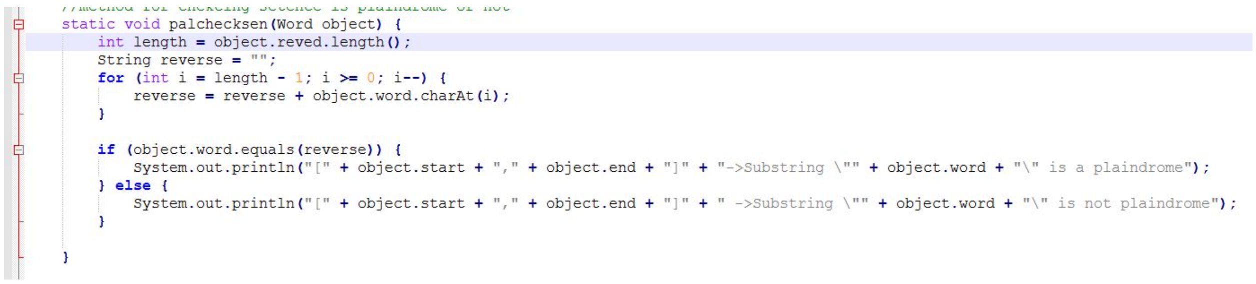
How to solve it ?

- Developed this algorithm using java.
- Firstly we have to give sentences and increasing word by word into the word class.
- Increase the word by word and you can checking every subqueries of string that can be generated by given sentence is palindrome or not.
- Where the index of the character in the string and table size is twice the size of the string.



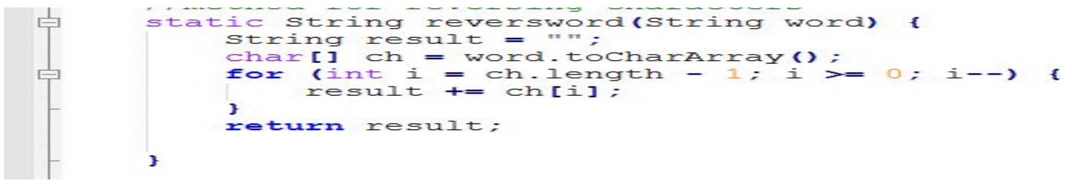
```java
for (int i = 0; i < cutsen.length; i++) {
    a[i] = new Word();
    a[i].word = cutsen[i];
    if (i == 0) {
        a[i].start = 0;
        a[i].end = cutsen[i].length() - 1;
    } else {
        a[i].start = a[i - 1].end + 2;
        a[i].end = a[i].start + cutsen[i].length() - 1;
    }
}
System.out.println("Checking each word of given sentence is palindrome or not ");

for (int i = 0; i < cutsen.length; i++) {
    palcheck(a[i]);
}
System.out.println("*****************************************************************************");
System.out.println("Checking every subquires of string that can be genrated by given setence is palindrome or not");
for (int i = 0; i < cutsen.length; i++) {
    Word concat = new Word();
    concat.word = a[i].word;
    concat.reved = reversword(a[i].word);
    for (int j = i + 1; j < cutsen.length; j++) {
        concat.word += " " + a[j].word;
        concat.reved += " " + reversword(a[j].word);
        concat.start = a[i].start;
        concat.end = a[j].end;
        palchecksen(concat);
    }

}
```

- Method for checking sentence is Palindrome or not.
- The palindrome check done by checking whether the hash values of characters in the subquery are available in the table.
- If you can check the word object is equal to reverse,the sentence is a palindrome and the word object is not equal to reverse , the sentence is not a palindrome.



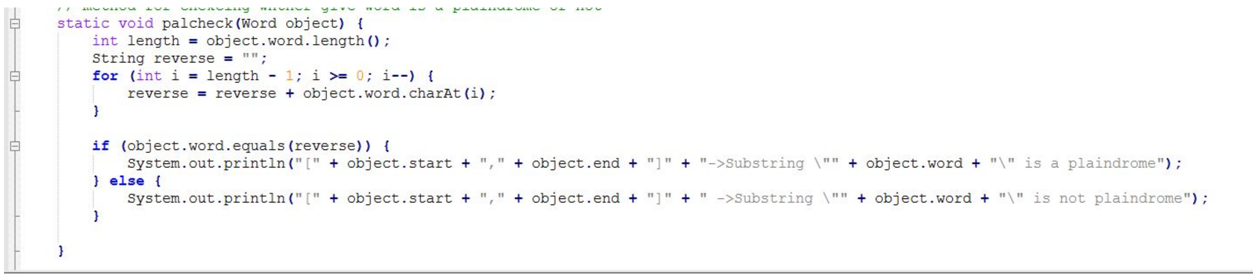
```java
static void palchecksen(Word object) {
    int length = object.reved.length();
    String reverse = "";
    for (int i = length - 1; i >= 0; i--) {
        reverse = reverse + object.word.charAt(i);
    }

    if (object.word.equals(reverse)) {
        System.out.println("[" + object.start + "," + object.end + "]" + "->Substring \"" + object.word + "\" is a plaindrome");
    } else {
        System.out.println("[" + object.start + "," + object.end + "]" + " ->Substring \"" + object.word + "\" is not plaindrome");
    }

}
```

- Method for reversing characters.
- If we use the static String reversword , you can check the reverse word objects.



```java
static String reversword(String word) {
    String result = "";
    char[] ch = word.toCharArray();
    for (int i = ch.length - 1; i >= 0; i--) {
        result += ch[i];
    }
    return result;

}
```

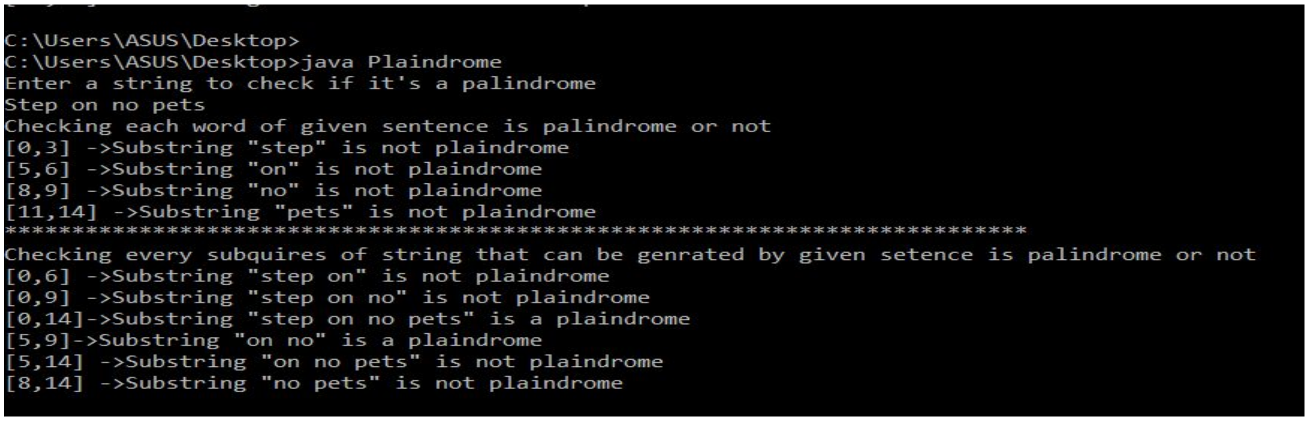
- Method for checking whether give is a palindrome.
- If the word object word is equal to reverse , the sentence is a palindrome.
- If the word object word is not equal to reverse , the sentence is not a palindrome.



```java
static void palcheck(Word object) {
    int length = object.word.length();
    String reverse = "";
    for (int i = length - 1; i >= 0; i--) {
        reverse = reverse + object.word.charAt(i);
    }

    if (object.word.equals(reverse)) {
        System.out.println("[" + object.start + "," + object.end + "]" + "->Substring \"" + object.word + "\" is a plaindrome");
    } else {
        System.out.println("[" + object.start + "," + object.end + "]" + " ->Substring \"" + object.word + "\" is not plaindrome");
    }

}
```

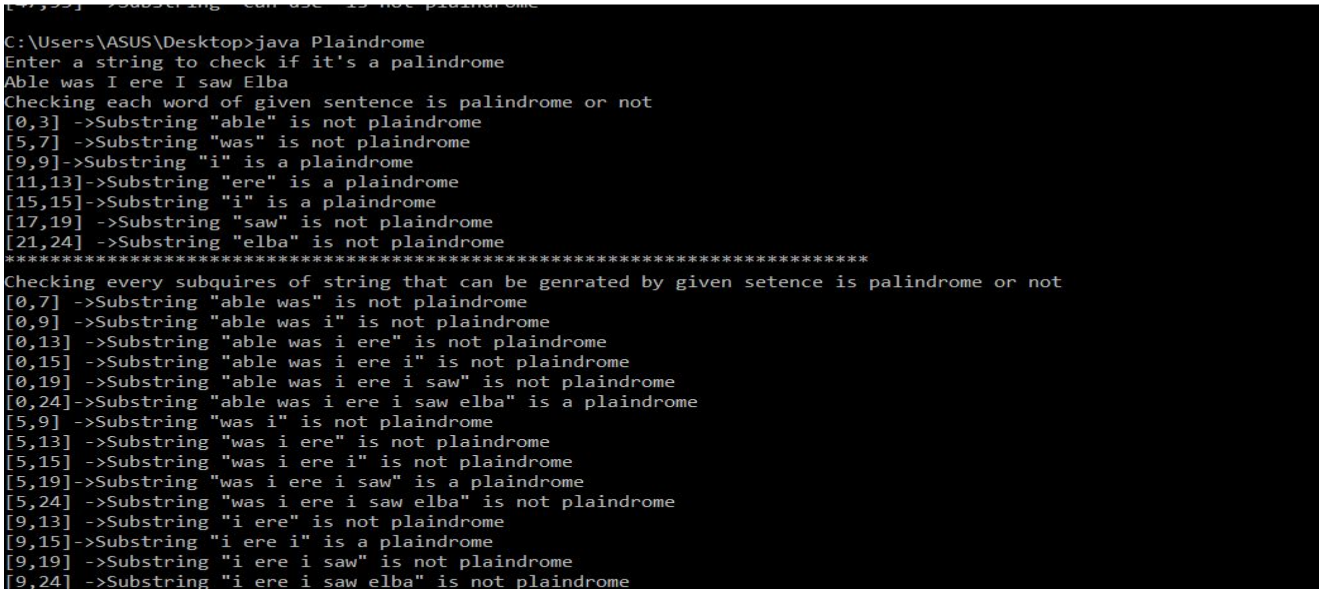
Output :

Output for the given scenario



```
C:\Users\ASUS\Desktop>
C:\Users\ASUS\Desktop>java Plaindrome
Enter a string to check if it's a palindrome
Step on no pets
Checking each word of given sentence is palindrome or not
[0,3] ->Substring "step" is not plaindrome
[5,6] ->Substring "on" is not plaindrome
[8,9] ->Substring "no" is not plaindrome
[11,14] ->Substring "pets" is not plaindrome
*****************************************************************************
Checking every subquires of string that can be genrated by given setence is palindrome or not
[0,6] ->Substring "step on" is not plaindrome
[0,9] ->Substring "step on no" is not plaindrome
[0,14]->Substring "step on no pets" is a plaindrome
[5,9]->Substring "on no" is a plaindrome
[5,14] ->Substring "on no pets" is not plaindrome
[8,14] ->Substring "no pets" is not plaindrome
```

Output for another test case



```
C:\Users\ASUS\Desktop>java Plaindrome
Enter a string to check if it's a palindrome
Able was I ere I saw Elba
Checking each word of given sentence is palindrome or not
[0,3] ->Substring "able" is not plaindrome
[5,7] ->Substring "was" is not plaindrome
[9,9]->Substring "i" is a plaindrome
[11,13]->Substring "ere" is a plaindrome
[15,15]->Substring "i" is a plaindrome
[17,19] ->Substring "saw" is not plaindrome
[21,24] ->Substring "elba" is not plaindrome
*****************************************************************************
Checking every subquires of string that can be genrated by given setence is palindrome or not
[0,7] ->Substring "able was" is not plaindrome
[0,9] ->Substring "able was i" is not plaindrome
[0,13] ->Substring "able was i ere" is not plaindrome
[0,15] ->Substring "able was i ere i" is not plaindrome
[0,19] ->Substring "able was i ere i saw" is not plaindrome
[0,24]->Substring "able was i ere i saw elba" is a plaindrome
[5,9] ->Substring "was i" is not plaindrome
[5,13] ->Substring "was i ere" is not plaindrome
[5,15] ->Substring "was i ere i" is not plaindrome
[5,19]->Substring "was i ere i saw" is a plaindrome
[5,24] ->Substring "was i ere i saw elba" is not plaindrome
[9,13] ->Substring "i ere" is not plaindrome
[9,15]->Substring "i ere i" is a plaindrome
[9,19] ->Substring "i ere i saw" is not plaindrome
[9,24] ->Substring "i ere i saw elba" is not plaindrome
```

.