**CSE 3038 PROJECT-1**

**Introduction**

In this project we are required to implement some algorithms using assembly (mips) in order to learn the structure of computers and learn assembly language. In this report we are going to explain algorithms that we use for project questions.

**Q1**

The first questions asks for finding longest palindrome for given input. A palindrome is basically a mirror for letters. This means that in input there should be even number of occurancy for each letter. Because the logic of mirror is basically having even number of thing. For example lets say the input is “aab” than the longest palindrome should be “aa” because there cannot be a reflection for letter b.

Lets continue with algorithm that we implemented. Before start for algorithm we would like to clarify that since q1 was our first attempt it is more complex than other questions and longer than each of that so some process may look like unnecessary.

Step-1

First we converted the input into lowercase if there is any uppercase in order to set all letters to same position (beacuse the ascii value of “A” and “a” is different)

Step-2

In the second step we calculated number of occurancy for each letter and stored it into an array. We used nested loop instruction in order to find number of occurancy. To understand te step let us show visually.

Input:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | a | a | b | c |

Number of occurancy:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 2 | 3 | 3 | 2 | 1 |

As we know each char use 1 byte and each int use 4 byte so we cannot store them consecutively therefore we stored them in index based different arrays. (first char of array has first int of occurancy)

Step-3

In the third step we removed redundant char from array and parse it to new array. Since characters occured once cannot be inside palindrome.

Step-4

In the fourth step we removed duplicated chars from array and parse it to new array. For the last step we should remove duplicate chars beacuse we are going to print each of them according to their occurancy and we did not want to print duplicated chars more than one time.

Step-5

In the fifth step we divided all occurancy by two and save the result at the same place that they were using floring division. (3/2=1.5)

Steep-6

In the last step we printed each char according to their new occurancy number. Than we printed each char again according to occurancy number but in the second time we started from back.(mirror effect)

**Q-2**

In this question we were supposed to replace vowels of given input. (“hello” to “holle”).

Step-1

Firstly we scanned the input for in order to check the vowels.

Step-2

After chechking the vowels inside input we put each vowels into new array.

Input:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| h | e | l | l | o |

Vowels:

|  |  |
| --- | --- |
| e | o |

Step-3

For the last step we started scanning input starting from last element beacuse we are going to replace each vowels of input with vowels inside vowels array. (replacing o with e and replacing e with o) and lastyl print result.

**Q-3**

For the third question there will be a int input and we are going to find if it is square-free or not. Square free means that if the multiplication of each distinct prime divider is uqual with input than it is square free number.

In this part we looped from 1 until given input (let say i value) If there is any i value that divides the input than we checked if it is prime. If it is prime than multiply it with constant register with value 1. When i reaches the input value we checked if number of dividers is one or not. If it is one than the input is prime. If it is not one than we checked if number hat we multiplied is equal with input or not. If it is equal than it square free and it is not square free otherwise.

We also stored prime number dividers in order to print all of them in the last part of q3.

**Notes:**

We could not do the q-4 and we could not implement q-1 inside menu. However there is menu for q-2 and q-3.