MARMARA UNIVERSITY

FACULTY OF ENGINEERING

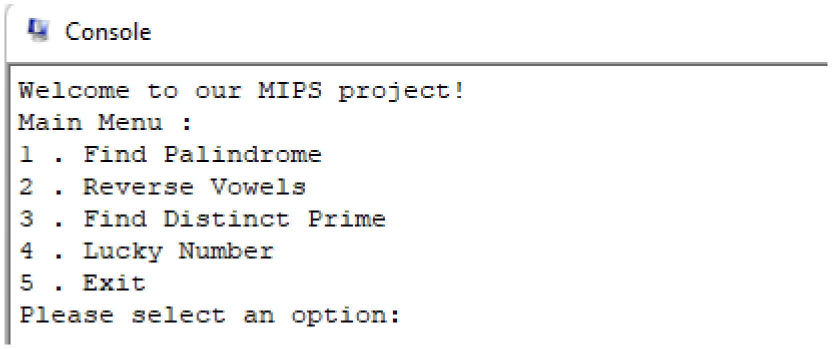
COMPUTER ENGINEERING DEPARTMENT

CSE3038 COMPUTER ORGANIZATION PROJECT-1

Number	First Name	Last Name
150119566	Müslim	Yılmaz
150119037	Ömer	Kibar
150119036	Serkan	Korkut
150119804	Berkay	Ağar

1- Main Function

Main function consists of infinite while loop that shows the menu to user and asks user to its choice in each iteration. According to user's choice it gets required input parameters and put them into argument registers and call corresponding functions. If user enter 5 while loop ends, and program terminates.



```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

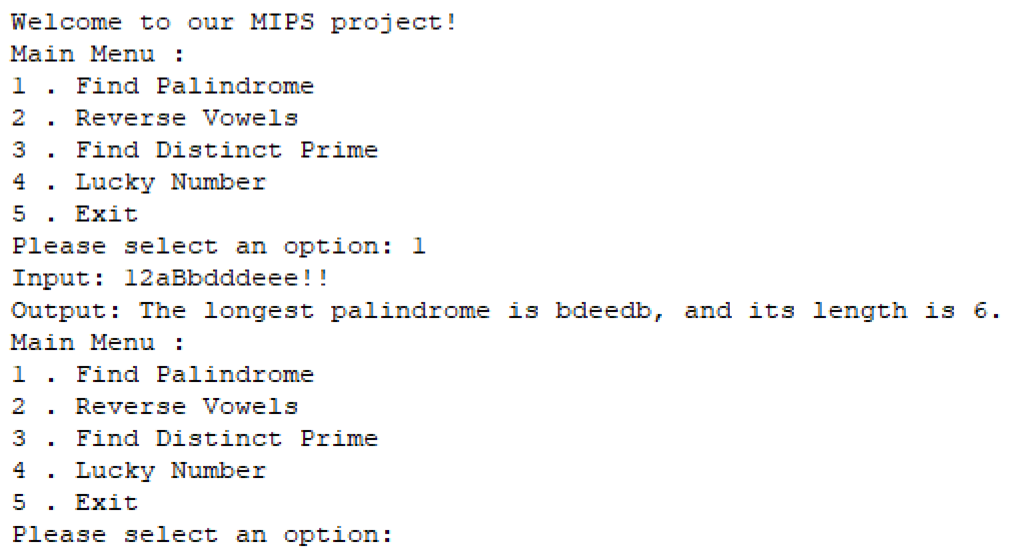
2- Find Palindrome

findPalindrome function is called from the main function with the one argument which is input string. The given input address is stored in $a0 register. First, the function loads the first character of the input and calls isLetter function. Here it first checks if the character is in the range a(97)-z(122). If not, this time it checks whether it is in the range of A(65)-Z(90). It converts it to lowercase by adding 32 to the character and returns. If the character is not a letter, it moves to the next character and checks if it is a letter. After it returns, it loads the same character in the other register and jumps to the notFinal label. Here it first checks if the character is null character. Secondly, it checks whether two characters are the same. If they are not the same, it jumps to next label. Here it replaces the second character with the last loaded one, loads the next character on the first character and returns to notFinal label. Otherwise, the count + 1. If the count is not equal to 2, it jumps to continue label. Here it loads the next character on the first character and returns to notFinal label. Otherwise, it prints the character, stores it on the stack, resets the count, and the palindrome length is +1. Then it loads the next character on the first character and returns to notFinal label.

If the character is null, it jumps to the final label. From there it jumps to the printSecond label and prints the characters in the stack, that is, the second half of the palindrome. Finally, it returns to the final label, multiplies the palindrome length by 2, prints the output and returns to the main function.

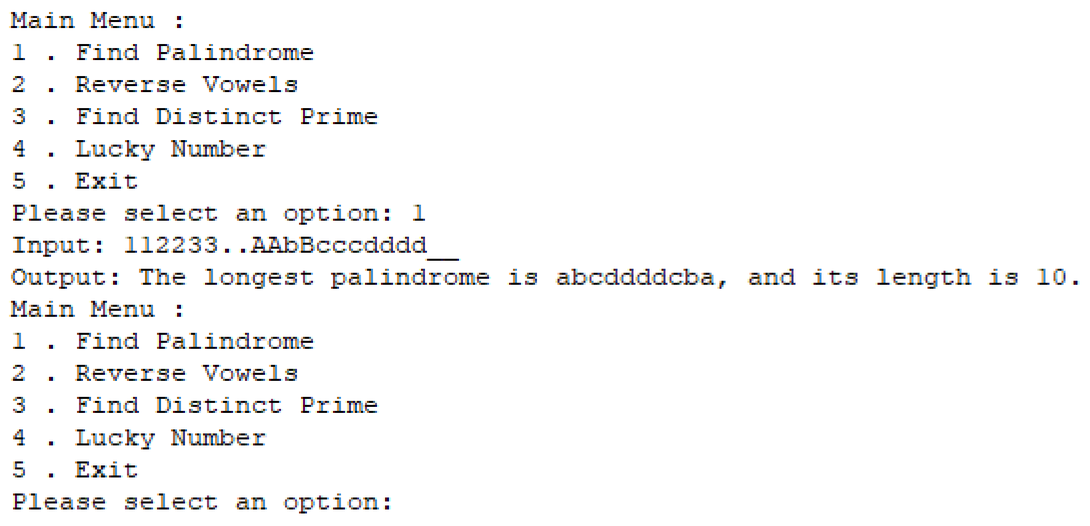
Example Runs;

Run 1.



```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 1
Input: 12aBbdddeee!!
Output: The longest palindrome is bdeedb, and its length is 6.
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

Run 2.



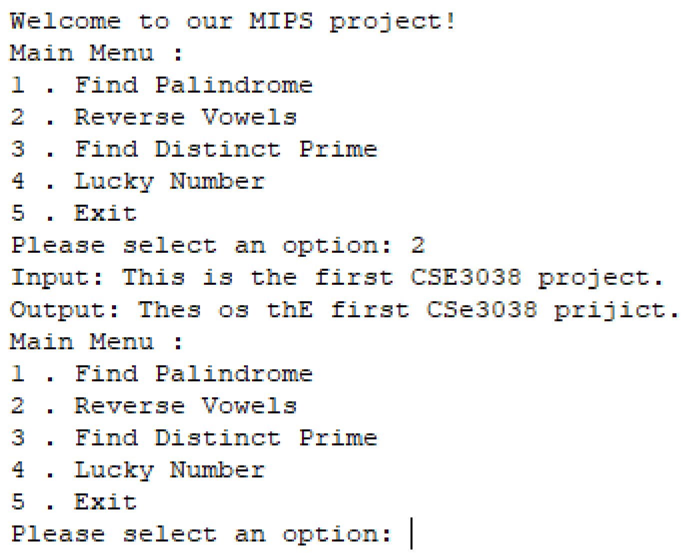
```
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 1
Input: 112233..AAbBcccdddd__
Output: The longest palindrome is abcddddcba, and its length is 10.
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

3- Reverse Vowels

reverseVowels function is called from the main function with the one argument which is input string. The given input address is stored in $a0 register. First it loads the first character and goes to gotoInputEnd label. Here it loads the next character continuously until the null character and reaches the end of the input. After that, it goes to L1 label. It loads characters from the beginning of the input and calls the isVowel function. In isVowel, checks if the character is one of the A, E, I, O, U, a, e, i, o, u characters. If true, it returns 1. If the return number is not 1, it jumps to the beginning of L1 and loads the next character. Otherwise, it jumps to L2 label. It loads characters starting from the end of input and calls the isVowel function. It jumps to swapChar label when the second vowel is found. In swapChar, if the address in L1 is not greater than or equal to the address in L2, it replaces 2 characters. Otherwise, it jumps to Last label, prints the output, and returns to the main function.

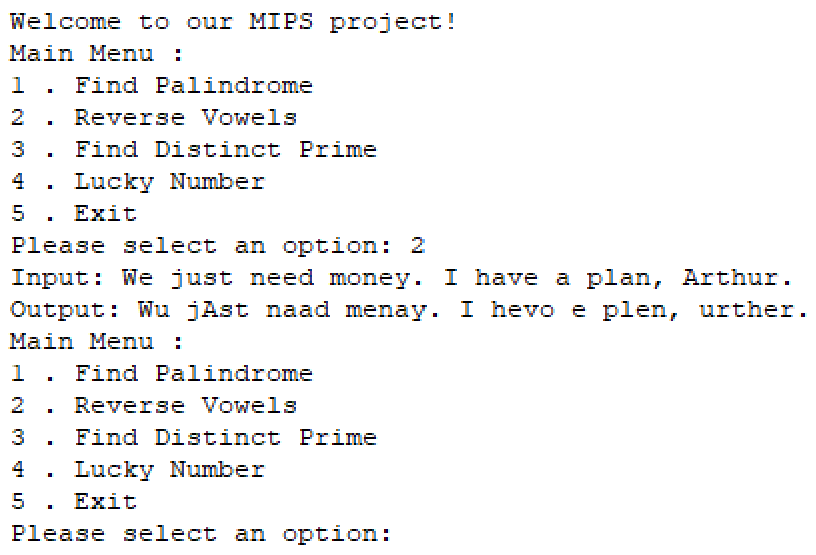
Example Runs;

Run 1.



```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 2
Input: This is the first CSE3038 project.
Output: Thes os thE first CSe3038 prijict.
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 
```

Run 2.



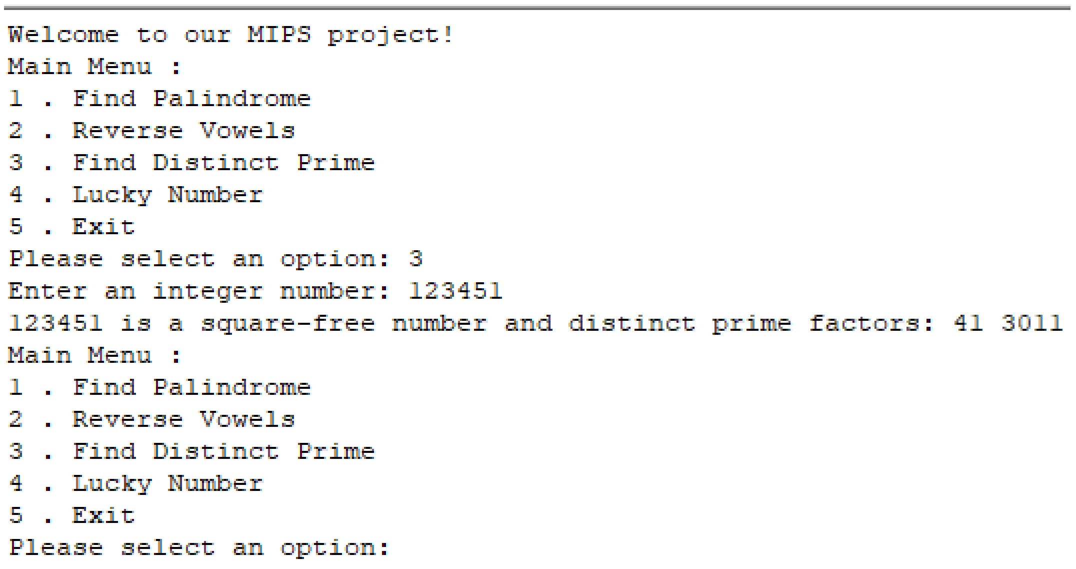
```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 2
Input: We just need money. I have a plan, Arthur.
Output: Wu jAst naad menay. I hevo e plen, urther.
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

4- Find Distinct Prime

findPrime function is called from the main function with the one argument which is input number. The given input number is stored in $a0 register. After necessary adjustment, findPrime function calls loopForSquareFree function which basically checks the i * i <= number condition in the while loop where i initially 2 and incremented by one at the end of each iteration. If condition false in some point, jump falseLabel label. falseLabel basically prints the given number and not square-free massage. Otherwise jump trueLabel. TrueLabel is first prints given input value. After printing, it start the findEachPrime process.This process basically calculates the unique primes and print them to the screen.There are several subprocesses under findEachPrime. InnerProcess label is divides given number accordingly and finds the prime factor numbers. keepDividingSameFactor Loop label is checks if, already given prime number is divides number or not. If true, just keep dividing same prime number. nextProcess is increment i by one and jump the loopForPrimes for same process. Lastly, exitFromLoopPrime is basically prints the space characters between each prime number and returns the $ra.

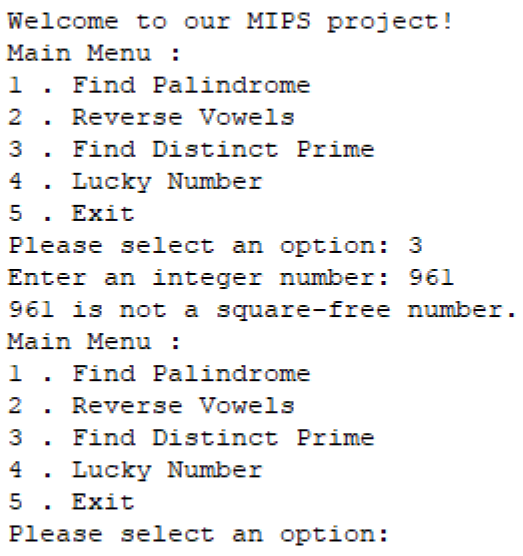
Example Runs;

Run 1.



```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 3
Enter an integer number: 123451
123451 is a square-free number and distinct prime factors: 41 3011
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

Run 2.



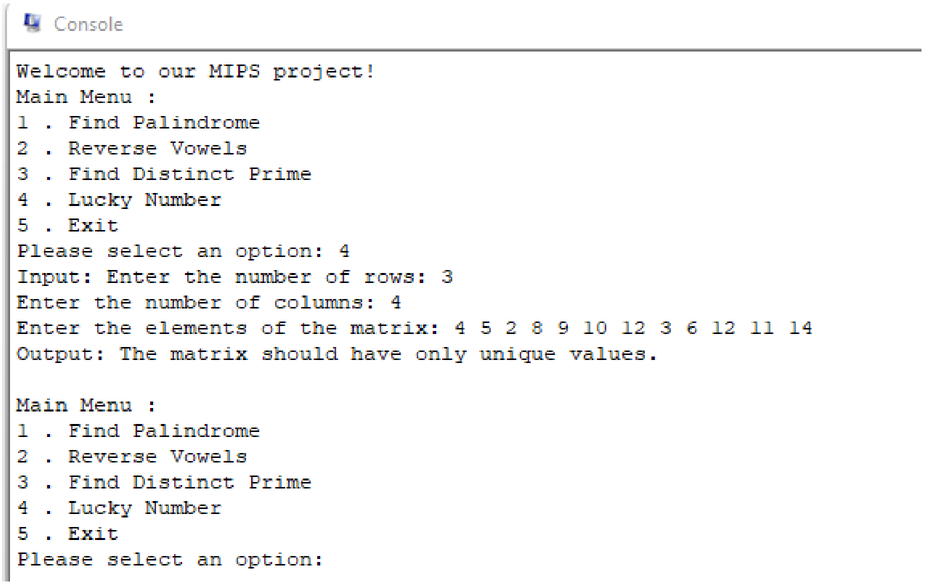
```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 3
Enter an integer number: 961
961 is not a square-free number.
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

5- Lucky Number

luckyNumber function is called from the main function with the arguments input string containing matrix elements, number of rows, number of columns in $a0, $a1, $a2 registers respectively. This function calls another function parseString which parses input string given in $a0 and converts it to an integer array and returns address of this integer array. Then luckyNumber function calls another function checkUniqueness which checks if the array given in $a0 contains unique elements or not. If the array is unique checkUnique function returns 0, otherwise it returns 1. After the return luckyNumber function checks the return value of checkUniqueness function. If return value is 1 it prints ‘The matrix should contain only unique values’ and returns to the main function. Otherwise, it starts to algorithm to find the lucky number. It first finds smallest in each row and store them in separate array. Then it finds greates in each column and store them in separate array. Then it finds the common element in these two arrays which is the lucky number in the matrix. Prints the lucky number and returns to the main function.

Example Runs;

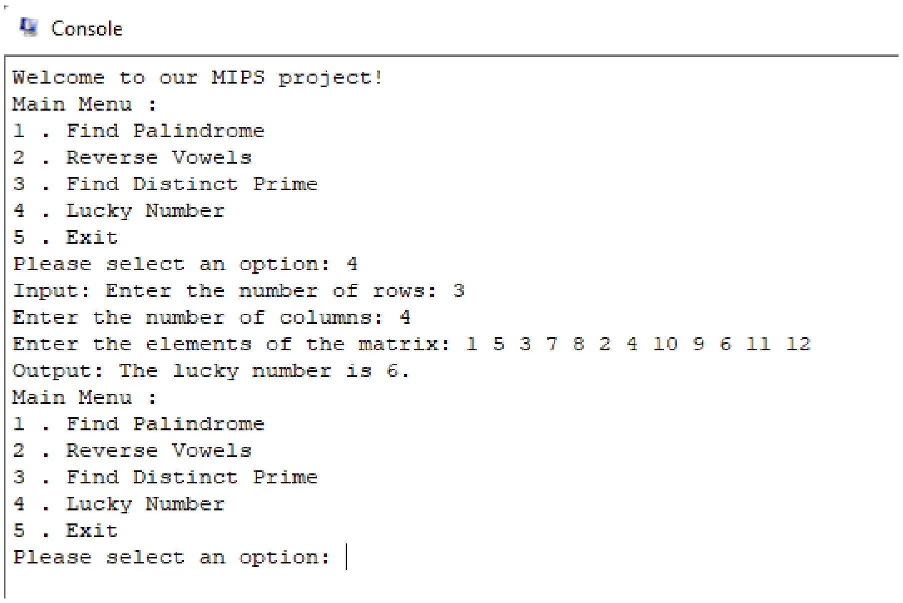
Run 1



```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 4
Input: Enter the number of rows: 3
Enter the number of columns: 4
Enter the elements of the matrix: 4 5 2 8 9 10 12 3 6 12 11 14
Output: The matrix should have only unique values.

Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```

Run 2


Console

```
Welcome to our MIPS project!
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option: 4
Input: Enter the number of rows: 3
Enter the number of columns: 4
Enter the elements of the matrix: 1 5 3 7 8 2 4 10 9 6 11 12
Output: The lucky number is 6.
Main Menu :
1 . Find Palindrome
2 . Reverse Vowels
3 . Find Distinct Prime
4 . Lucky Number
5 . Exit
Please select an option:
```