**ABSTRACT**

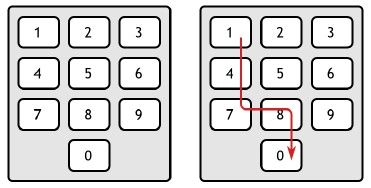
The proposed project helps us to set a correct timer if we have entered wrong timer. This proposed project finds the closest number to entered timer if it does not satisfies the conditions.

Once entered the number, then this project will first check whether your input (number i.e. timer) satisfies the condition or not. If it satisfies the condition, then it will get printed as it is. Otherwise, this project will find the possible numbers which satisfy the given condition and will the pick one number from them which will be the closest to the input and will print it .

**PROBLEM STATEMENT**

Find the closest number to entered timer if it does not satisfies the conditions.

**PROBLEM DESCRIPTION**



When the user set the timer, then this project will first check whether timer satisfies the condition or not.

If it satisfies the condition, then it will get printed as it is. Otherwise, this project will find the possible numbers which satisfy the given condition and will the pick one number from them which will be the closest to the input and will print it.

**REQUIREMENT SPECIFICATION**

1. To display the keyboard we need one matrix of dimension 4\*3.
2. That matrix will get stored in the form of directed graph. So, one adjacency matrix of dimension 10\*10 is needed. This graph contains only the downward and rightward edges.
3. The input timer will get stored in variable ‘k’.
4. The class ‘Microwave oven’ which contain 4\*3 matrix, 10\*10 adjacency matrix and timer ‘k’. So that these data members are easily accessible in all member functions.
5. Timer should be 1<= *timer*<=200.
6. You have to set a timer when our hand following down.
7. Once your hand goes down, it can’t go up to enter next number (digit in timer).
8. Similarly, once your hand goes on right side on the keyboard, it can’t go to the left side to enter next number (digit in timer).
9. Thus, you can move your hand only downward and rightward on the keyboard while entering the timer.
10. You can press the same key again and again on the keyboard.
11. Program must display number that is closest to timer which can be inputted by the user.

**REQUIREMENT ANALYSIS**

The keyboard in the form of matrix of dimension 4\*3.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| 10 | 0 | 10 |

Fig. a

In fig b. in the block of [4][1] and [4][3] we have place the values which are greater the 9 and they are not allow to use and they will not get printed in the keyboard.

**Adjacency Matrix of Graph**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Fig. b

In the above fig there are 10 rows starting from 0 to 9 and similarly 10 columns.

By looking graph, if the path present between the digits of the keyboard the place ‘1’ in that respective block ([r][c]). Otherwise place ‘0’.

Thus when user enters number, this separate each digit and whether the path present in between them or not and take further decisions.

**PROBLEM SOLUTION**

**Main algorithm:**

1. Get the numbers of test case from user.
2. If the numbers of test cases are zero then,

-show error message.

1. Else,
2. Create an array of those test cases.
3. Get input from user one by one and pass into an array.
4. Get number from array and check 1<=number<=200.
5. If yes,
6. Check whether it satisfies the condition or not.
7. If yes, then print that number as it is.
8. Otherwise find all possible number which are almost closer to it and find the number from them such that it will be the closest number to the input number.
9. If no, then show error message.
10. If array has not been ended, then go to step 3.

**Step** 3.4.1:

1. If 1<=number <=9

1. Print the number as it is.

1. If 10<=number<=99
2. Separate each digit in number as ‘D1’ and ‘D2’.
3. Check whether the path is present between D1 and D2 or not.
4. If yes, print the number as it is.
5. Else,
6. Keep the D2 digit as it is and finds the all digits on the keyboard nearer to digit D1such that, path is present in between D2 and those digits.
7. Create an array of size 10 e.g. arr1 [10] and pass that all digits into the array. When all digits are finished then pace 100 at that current index (i) of an array.
8. Do the step 2 until the value at current index (i) is not 100.
9. Create an array of size 10. E.garr2 [10].
10. Perform the subtraction between D1 and each element of an array.
11. If the array element is greater than D1 then, subtract D1 from that array element.
12. Else, subtract array element from D1
13. Pass each result into arr2 [].
14. When all results are finish then place 200 at current index (i) of arr2 [].
15. If the value at arr2 [i] is 200 then close the loop.
16. Sort the array arr2 [].
17. Add 0th element of arr2[], to input number and store it into another variable e.g. z.
18. Repeat the step 2.1, 2.2 and 2.3 for ‘z’.
19. Else, subtract the 0th element of arr2[] from input number and print it.
20. If 100<=number<=200
21. Separate each digit as ‘D1’,’D2’ and ‘D3’
22. Check whether the path is present in between D3 and D2, (there must be a path present between the D3 and D2 because the number<=200)
23. If yes, then check whether the path is present in between D2 and D1
    1. If yes, then print the input number as it is.
    2. Else,
24. Keep the D3 and D2 as it is and repeat the step 2.4.1 to 2.4.5.9 for D1.

**FLOWCHARTS**

Get input from one by one and pass into an array

Create and array of that test cases

Else

Show error message

if number of test cases are zero

Get the number of test cases

START

Get number from from array and check 1<=number<=200

If yes

Print that number as it is

Otherwise find all possible numbers are almost closer to it and from them such that

It will print closest number to the input number

If no

If 10<=number <=99

Print the number as it is

If 1<=number<=9

If array has not been ended then go to step 3

Show error message

Show error message

If array has not been ended then go to step 3

If 1<=number<=9

Print the number as it is

If 10<=number <=99

Separate each digit in number as ‘D1’ and ‘D2’

Check whether the path is present between D1

And D2 or not

If yes

Print the number as it is

else

Keep D2 as it is and find D1 nearer digit such that, path is present in between D2 and that digits

Create an array of size of 10 i.e. arr1 [10] and find pass that all digits into the array. When all digits are ended place 100 at current index (i) of an array

Do step 2 until the value at current index(i) is not 100

Create an array of size 10 i.e.arr2[10]

Perform the subtraction between D1 and each of an array

If array element<D1 then, subtract D1 from that array element

Else, subtract array element from D1

Pass each result to arr2 []

When all result ends add 200 at current index() of arr2[]

If the value of arr2 [i] is 200 then close the loop

Sort the array arr2 []

Add 0th element of arr2 [], to input number and store it into another variable i.e. z

Repeat the step 2.1,2.2 and 2.3 for ‘z’

Else, subtract the 0th element of arr2[] from input number and print it

If 10<=number<=200

Check whether the path is present in between the D3 and D2, (there must be path exist between D2 and D2 because the number <=200)

If yes

Then check whether the path is present in between D2 and D1

If yes

else

Keep the D3 and D2 as it is and repeat the step 2.4.1 to 2.5.9 for D1

Stop

**IMPLEMENTATION**

Functions:

**creatKeyboard():**

It actually creates a 4\*3 matrix and store the number in that matrix according to given format of keyboard.

**adjMat():**

It creates and adjacency matrix of dim. 10\*10 e.g. adj[10][10]. It calls the function checkIndex() whatever value will be return, will be store in variable flag.

If flag=1, then it places ‘1’ at current ‘r’ and ‘c’ i.e. adj[r][c]=1.

Else,adj[r][c] = 0,

Where r = row and c = column.

**checkIndex():**

It gets two inputs from function adjMat(),

e.g. num1 and num2

if column or row of num2 > the column or row of num1

then, it returns 0

else, it returns 1.

**showKeyboard(void):**

It call the function creatKeyboard().

They it displays the keyboard and then it calls the function adjMat() to create adjacency matrix.

**checkNumber():**

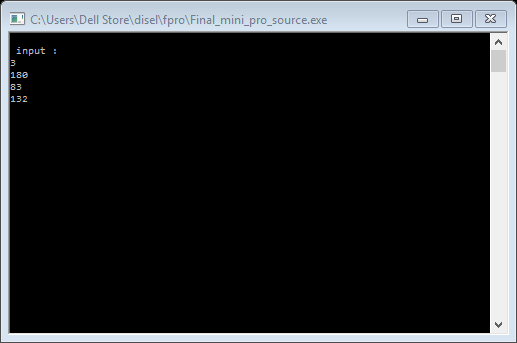
This function take input from source code. It check whether the number is satisfies the condition or not and calls the function findNumbers().

**findNumbers():**

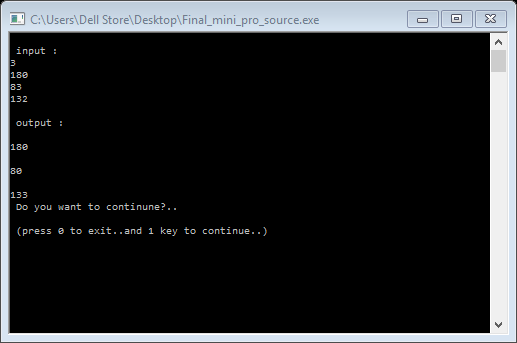
It takes two input from function checkNumber() and follows the steps in algorithm, from 2.4.1 to 3.1.2 in **Step 3.**4.1

**SNAP SHOT**

**INPUTS:**

****

**OUTPUTS :**

****

**CONCLUSION**

The proposed project calculate the correct timer if the user enters wrong timer. This project helps to user to set the timer automatically if the timer is wrong so that user can heat the milk for correct timer and so he can enjoy the coffee.

**REFERENCES**

Books:

1. An Integrated Approach To Software Engineering

By PankajJalote.

1. Lets us C++, by Y. Kanetkar