Basics of Programming Assignment

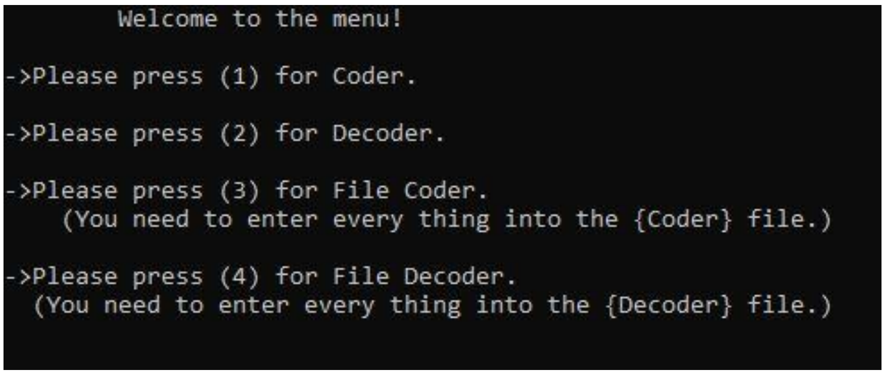
“Morse Code Converter”

Developer: Turker Erbasi

Neptun Code: NVT5RE

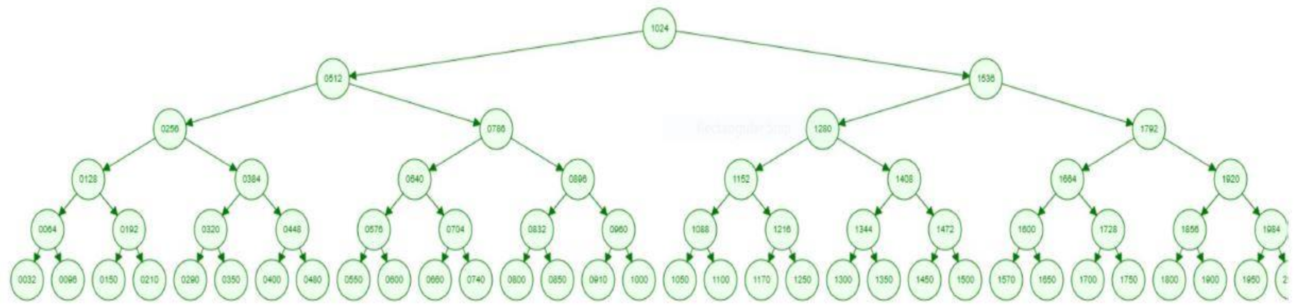
1. Description and User Guide of the project:

   The User chooses what to do with the program from “Menu” section. If user choose “1”, the program will take user to the “Coder” section. In this section anything that user types will be translated to the Morse code after he/she hits the “enter” button. If user chooses the “2”, program will take user to the “Decoder” section.
   In this section anything that user types will be translated to the Morse code after user hits “enter”. If user chooses the option “3”, program takes user to the “Coder” section again, although, this time it will require a “.txt” file named as “Coder”. Then this program will read that file and translate it into Morse code. For the option “4” it will work as the reverse of the option “3”.



```
        Welcome to the menu!

->Please press (1) for Coder.

->Please press (2) for Decoder.

->Please press (3) for File Coder.
    (You need to enter every thing into the {Coder} file.)

->Please press (4) for File Decoder.
  (You need to enter every thing into the {Decoder} file.)
```

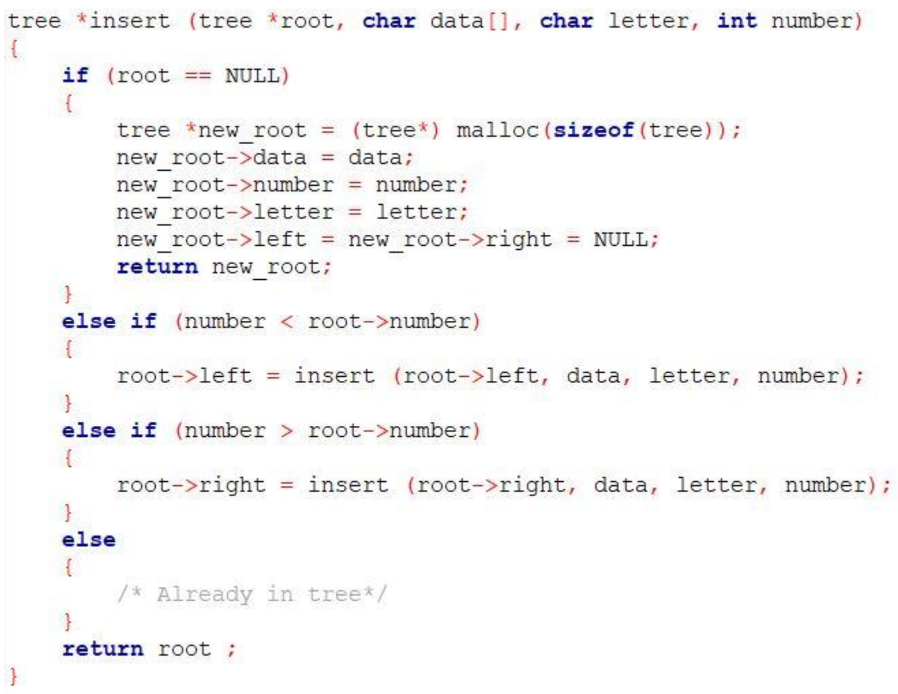
2. Tree Build and Visualization of Binary Tree of the Project:



On purpose of building the tree - program needs a “insert” function. With this function the program can build every single node (root) with the help of the for loop.

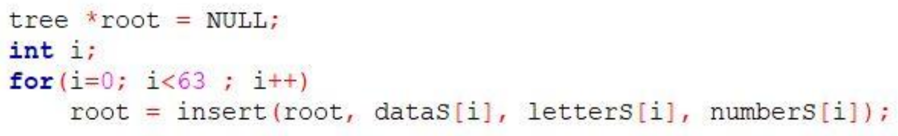
Building the Tree:

In order to build the tree, I used the insert function. *(screen shot below)*
In this function: if the root is NULL, it allocates new memory for new node and fill it.



```c
tree *insert (tree *root, char data[], char letter, int number)
{
    if (root == NULL)
    {
        tree *new_root = (tree*) malloc(sizeof(tree));
        new_root->data = data;
        new_root->number = number;
        new_root->letter = letter;
        new_root->left = new_root->right = NULL;
        return new_root;
    }
    else if (number < root->number)
    {
        root->left = insert (root->left, data, letter, number);
    }
    else if (number > root->number)
    {
        root->right = insert (root->right, data, letter, number);
    }
    else
    {
        /* Already in tree*/
    }
    return root ;
}
```

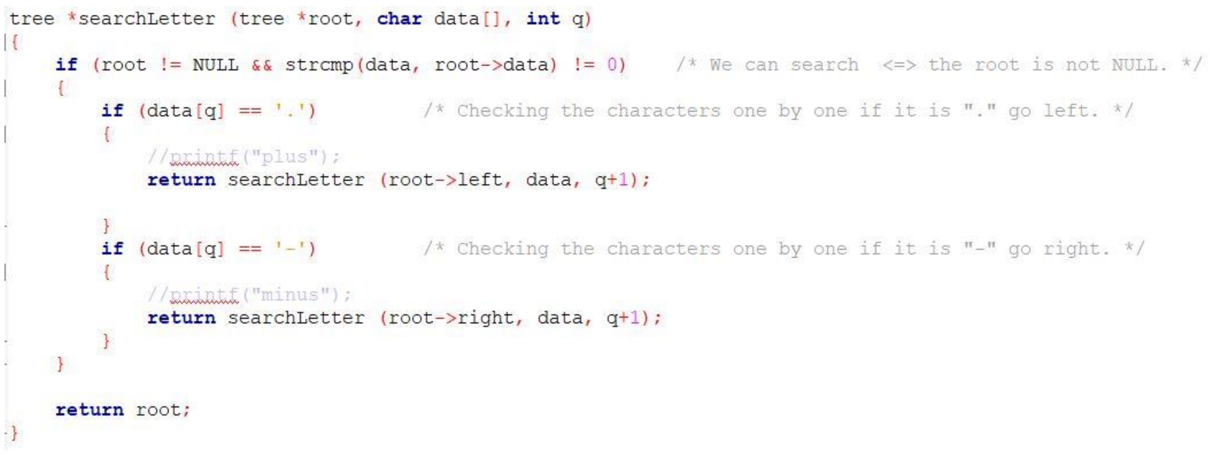
To fill it, there is an auxiliary “for” loop for it. Repeats 63 times. (we have 63 nodes in the tree)



```c
tree *root = NULL;
int i;
for(i=0; i<63 ; i++)
    root = insert(root, dataS[i], letterS[i], numberS[i]);
```

3. Search function:

Search function takes the strings which are we cut it out for Morse letters from user-input before and compare them with the first node. if it is not that node, it checks the first dot or dash to find its way recursively.

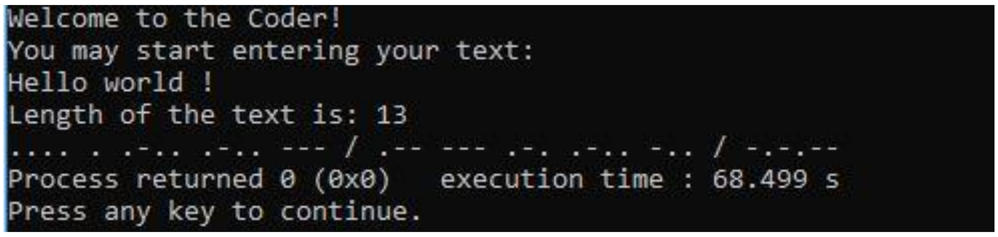


```c
tree *searchLetter (tree *root, char data[], int q)
{
    if (root != NULL && strcmp(data, root->data) != 0)     /* We can search  <=> the root is not NULL. */
    {
        if (data[q] == '.')          /* Checking the characters one by one if it is "." go left. */
        {
            //printf("plus");
            return searchLetter (root->left, data, q+1);

        }
        if (data[q] == '-')          /* Checking the characters one by one if it is "-" go right. */
        {
            //printf("minus");
            return searchLetter (root->right, data, q+1);
        }
    }

    return root;
}
```

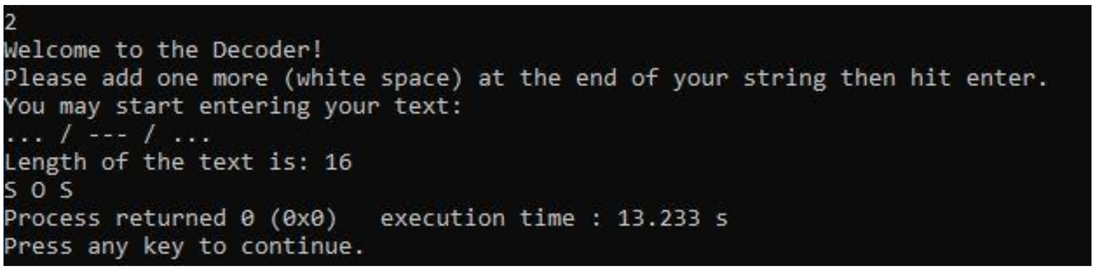
4. <span>Translation from a File:</span>

   For translation the program uses the same functions that we mentioned before. Section “3” and “4” open the file for reading and reads the string in the file. It calls the required function for translation (Search). Then prints the converted version on the screen.

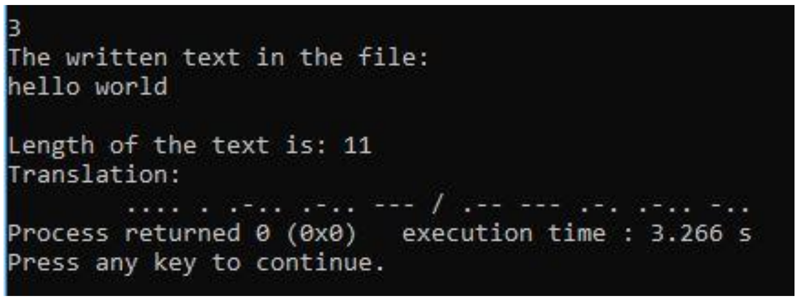
5. Some Screen Shots from Program:



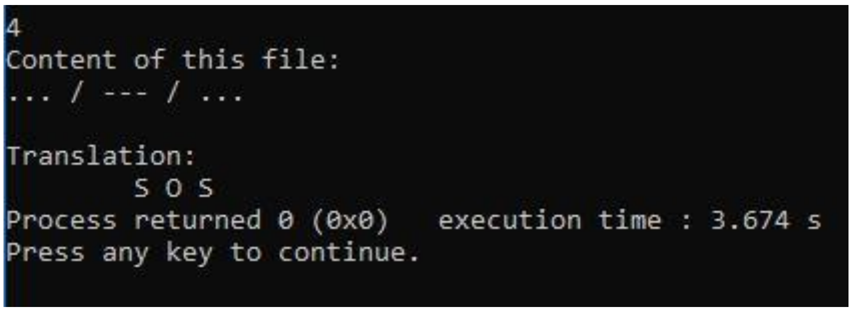
```
Welcome to the Coder!
You may start entering your text:
Hello world !
Length of the text is: 13
.... . .-.. .-.. --- / .-- --- .-. .-.. -.. / -.-.--
Process returned 0 (0x0)   execution time : 68.499 s
Press any key to continue.
```



```
2
Welcome to the Decoder!
Please add one more (white space) at the end of your string then hit enter.
You may start entering your text:
... / --- / ...
Length of the text is: 16
S O S
Process returned 0 (0x0)   execution time : 13.233 s
Press any key to continue.
```



```
3
The written text in the file:
hello world

Length of the text is: 11
Translation:
        .... . .-.. .-.. --- / .-- --- .-. .-.. -..
Process returned 0 (0x0)   execution time : 3.266 s
Press any key to continue.
```



```
4
Content of this file:
... / --- / ...

Translation:
        S O S
Process returned 0 (0x0)   execution time : 3.674 s
Press any key to continue.
```