

Executive Summary

This report is created based on the object-oriented programming with C++. Object-oriented programming is a paradigm which relies the concept of objects and classes. There have several types of object-oriented in the software development world, such as Java, C++ and so on. This report is created based on the C++ program. In this report an English dictionary is created by using C++ programming and Visual studio is used a platform of C++ execution. Part 1 and part 2 holds the description and code of the tasks which are done in the English dictionary, such as word meaning retrieve, parts of speech checking and so on.

Table of Contents

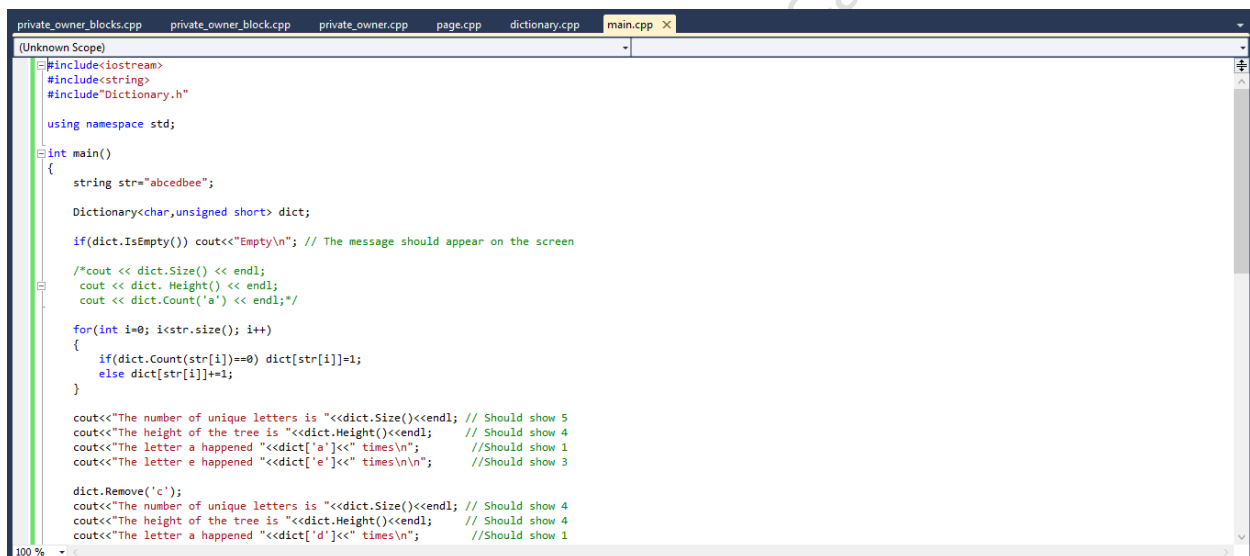
Introduction.....	3
Part 1	3
Part 2	8
Conclusion	12
References.....	13

Programming Assignment Sample By Call Assignment Help

Introduction

Object-oriented programming is a programming model which organizes the software design around the data or information or objects rather than logic and function. It focuses on the objects which are manipulated by the developers. Java, C++ is common example of object-oriented programming language. In this case, the researcher is going to discuss the C++ programming language. During this research, in part 1 and part 2 the researcher is going to solve some problem by using the C++ program. In order to do this job, Visual studio software is used by the researcher, which is very popular programming execution software. C++ and C# programming language are executed in this software.

Part 1



```
private_owner_blocks.cpp private_owner_block.cpp private_owner.cpp page.cpp dictionary.cpp main.cpp x
(Unknown Scope)
#include<iostream>
#include<string>
#include"Dictionary.h"

using namespace std;

int main()
{
    string str="abcedbee";

    Dictionary<char,unsigned short> dict;

    if(dict.IsEmpty()) cout<<"Empty\n"; // The message should appear on the screen

    /*cout << dict.Size() << endl;
    cout << dict.Height() << endl;
    cout << dict.Count('a') << endl;*/

    for(int i=0; i<str.size(); i++)
    {
        if(dict.Count(str[i])==0) dict[str[i]]=1;
        else dict[str[i]]+=1;
    }

    cout<<"The number of unique letters is "<<dict.Size()<<endl; // Should show 5
    cout<<"The height of the tree is "<<dict.Height()<<endl; // Should show 4
    cout<<"The letter a happened "<<dict['a']<<" times\n"; //Should show 1
    cout<<"The letter e happened "<<dict['e']<<" times\n\n"; //Should show 3

    dict.Remove('c');
    cout<<"The number of unique letters is "<<dict.Size()<<endl; // Should show 4
    cout<<"The height of the tree is "<<dict.Height()<<endl; // Should show 4
    cout<<"The letter a happened "<<dict['d']<<" times\n"; //Should show 1
```

Figure 1: Main file

(Source: Created using Microsoft Visual Studio)

The above-mentioned code is the code of the main file; this is the primary of a programming file. In C++, the main is the only source file, which helps the user to execute another file in a proper manner. In this case, the researcher created this file in order to produce a dictionary, and that is why the researcher has added a dictionary. h library in the main file. After the implementation of

the libraries, the main class is created by the researcher, which holds a food loop, if statement, subclasses (remove, balance, clear) and some cout commands.

Word class and Dictionary class correctly implemented

```
#include <algorithm>

namespace vega {
namespace dictionary {
    const std::string& Dictionary::default_dictionary_file_name() {
        static const std::string filename = "/usr/local/share/vega/dictionary.txt";
        return filename;
    }
}

void set_dictionary(const std::string& file_name) {
    Dictionary::set_dictionary(file_name);
}

const Dictionary& instance(bool allow_default) {
    return Dictionary::instance(allow_default);
}

std::unique_ptr<Dictionary> Dictionary::singleton_ = nullptr;

void Dictionary::set_dictionary(const std::string& file_name) {
    Dictionary::singleton_ = std::unique_ptr<Dictionary>(new Dictionary(file_name));
}

const Dictionary& Dictionary::instance(bool allow_default) {
    if (!Dictionary::singleton_) {
        // If allowing default dictionary, then can initialize it now
        if (allow_default) {
            set_dictionary(default_dictionary_file_name());
        }

        // If still no dictionary found, then raise error
        if (!Dictionary::singleton_) {
            throw vega::Exception("Dictionary not yet initialized with call to set_dictionary()");
        }
    }
}
```

Figure 2: Implementation of the dictionary class

(Source: Created using Microsoft Visual Studio)

This image visualized the code of the implementation of the dictionary class. This class is used to create the English dictionary by using the C++ programming language. Under the class, the researcher has used the Const and stat const declaration for the deceleration of the functions. In C++ const declaration is used to create a read-only reference to a value, which means the identifier of the value is cannot be reassigned. This declaration is also used in pointer declaration. After it, the researcher has used different subclasses, operators and conditions one by one.

After the execution of this code, if the system creates a default dictionary, then the user can initialize that dictionary easily, but in case this code cannot allow the system to used the default dictionary than it visualized an error in the screen. For doing this job, the if-else statement is used by the researcher.

Dictionary file loaded and parsed correctly

```
if (std::regex_match(line, line_match, public_page_regex)) {
    std::string group_s = line_match[1].str();
    std::string element_s = line_match[2].str();

    const TagMask tag_mask(group_s, element_s);
    const MultiVR multi_vr(line_match[3].str());
    const std::string name = line_match[4].str();
    const VM vm(line_match[5].str());

    auto page = std::make_shared<const Page>(name, tag_mask, multi_vr, vm);
    pages.push_back(page);
}
else if (std::regex_match(line, line_match, private_page_regex)) {
    auto it = m_name_to_private_owner.find(line_match[1].str());
    std::shared_ptr<PrivateOwner> private_owner;

    if (it == m_name_to_private_owner.end()) {
        private_owner = std::make_shared<PrivateOwner>(line_match[1].str());
        m_name_to_private_owner.emplace(private_owner->name(), private_owner);
    }
    else {
        private_owner = it->second;
    }

    const TagMask tag_mask(line_match[2].str(), line_match[3].str());
    // FIXME: Allow invalid private tags?
    if (tag_mask.value_tag().group() < 9) continue;
    // FIXME: Allow group masks like 60xx and 70xx
    if (tag_mask.mask_tag() != Tag{0xFFFF,0x00FF}) continue;
    // NOTE: Why the heck are there "private" elements in gdcn with even groups?
    if ((tag_mask.value_tag().group() & 1) == 0) continue;
}
```

Figure 3: Implementation of different things in the dictionary

(Source: Created using Microsoft Visual Studio)

In case the code of figure 2 is executed in a proper manner and allows the system to use the default dictionary, then the user can implement different symbols, numbers in the default dictionary by using the code of figure number 3. In order to do this job, the researcher has used a while loop and under the while loop the researcher has to implement some if-else statement. In C++ language while loop is used to repeat a block of code several time until the condition becomes matched (Alfianto *et al.* 2017, p. 012043). Each if-else statement allows the user to implement different symbol special characters and numbers in the dictionary. In this case, the researcher has used the while loop in order to execute a condition fir several times.

```

// Generic private page
auto page = std::make_shared<const Page>(std::string("PrivateCreator"), TagMask{0x0001, 0x0001, 0x0000, 0x0000}, MultiVR{"LO"}, VM{1, VM::MAX_LIMIT});
pages.push_back(page);
}
this->add_pages(pages);
}

void Dictionary::add_pages(const std::vector<std::shared_ptr<const Page>>& pages) {
    std::set<unsigned> popcounts;

    unsigned popcount;

    for (const auto& page : pages) {
        m_name_to_page.emplace(page->name(), page);

        popcount = page->tag_mask().mask_popcount();
        popcounts.insert(popcount);

        {
            auto it = m_popcount_to_tag_masks.find(popcount);
            if (it == m_popcount_to_tag_masks.end()) {
                m_popcount_to_tag_masks.emplace(popcount, std::set<Tag>());
                it = m_popcount_to_tag_masks.find(popcount);
            }

            it->second.insert(page->tag_mask().mask_tag());
        }

        {
            auto it = m_mask_tag_to_map_from_value_tag_to_pages.find(page->tag_mask().mask_tag());
            if (it == m_mask_tag_to_map_from_value_tag_to_pages.end()) {
                m_mask_tag_to_map_from_value_tag_to_pages.emplace(page->tag_mask().mask_tag(), std::map<Tag, std::shared_ptr<const Page>>());
                it = m_mask_tag_to_map_from_value_tag_to_pages.find(page->tag_mask().mask_tag());
            }
        }
    }
}

```

Figure 4: Creation of private page

(Source: Created using Microsoft Visual Studio)

After creating the dictionary the researcher has done some test with the dictionary in order to check whether the dictionary is properly working or not. At the first, the researcher has tried to implement different symbol, number, and special character in the dictionary, which is visualized in figure 5 and the code is executed properly. After the successfulness of the symbol implementation, the researcher has tried to create a private page in the dictionary, which is only accessible by the researcher. For the code of figure number 4, no outsider can visualize this private page. Because of this code the user has to give a userid and password in the system in order to access this page, in case the user do not give the correct Userid or password than a popup page become arise and visualized a message for retry.

Basic Tasks

```

CTernarySearchTree::CTernarySearchTree()
{
    root = NULL;
    originalstring = NULL;
    srctop = 0;
    number_of_words = 0;
}

CTernarySearchTree::~CTernarySearchTree()
{
    strList.RemoveAll();
}

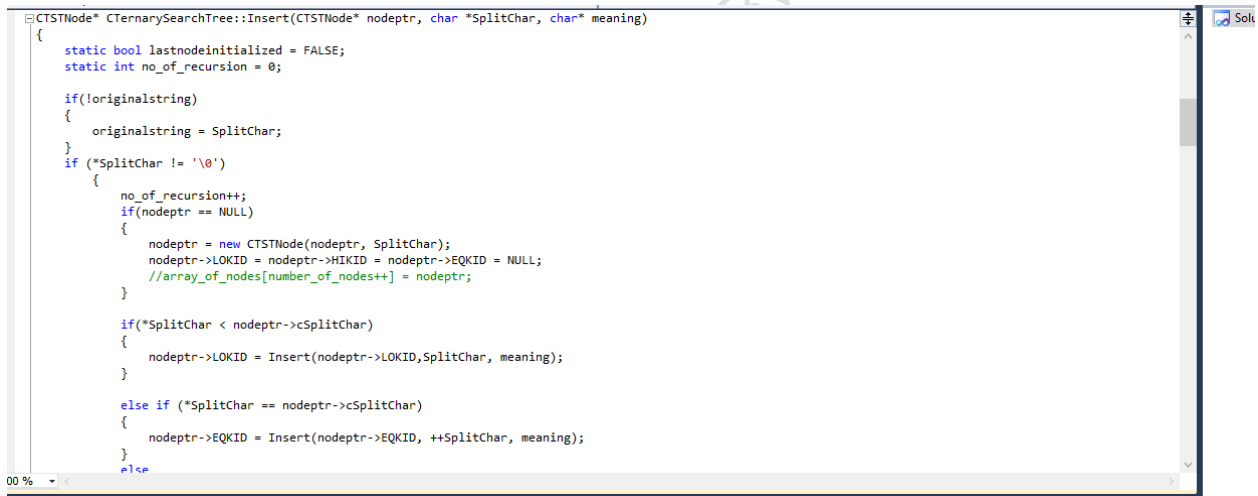
```

Figure 5: Code of task 1

(Source: Created using Microsoft Visual Studio)

In order to solve task 1, the researcher has used the upper mentioned C++ code. This helps the researcher to find all the required data of a word, such as meaning, parts of speech and many other things. In any case, this code is unable to find the required data or cannot match user given data with the dictionary stored data than it visualized a popup message box, which contains a “data is not found” message. During the execution of this programming, the researcher has tried to find the meaning of the phone and he or she get the required data in a proper manner.

In order to create or execute this code in a proper manner, the researcher has used an if-else statement several times to execute some conditions. The main job of this code is to match the user given data with the sorted dictionary data.



```
CTSTNode* CTernarySearchTree::Insert(CTSTNode* nodeptr, char *SplitChar, char* meaning)
{
    static bool lastnodeinitialized = FALSE;
    static int no_of_recursion = 0;

    if(!originalstring)
    {
        originalstring = SplitChar;
    }
    if (*SplitChar != '\0')
    {
        no_of_recursion++;
        if(nodeptr == NULL)
        {
            nodeptr = new CTSTNode(nodeptr, SplitChar);
            nodeptr->LOKID = nodeptr->HIKID = nodeptr->EQKID = NULL;
            //array_of_nodes[number_of_nodes++] = nodeptr;
        }

        if(*SplitChar < nodeptr->cSplitChar)
        {
            nodeptr->LOKID = Insert(nodeptr->LOKID, SplitChar, meaning);
        }

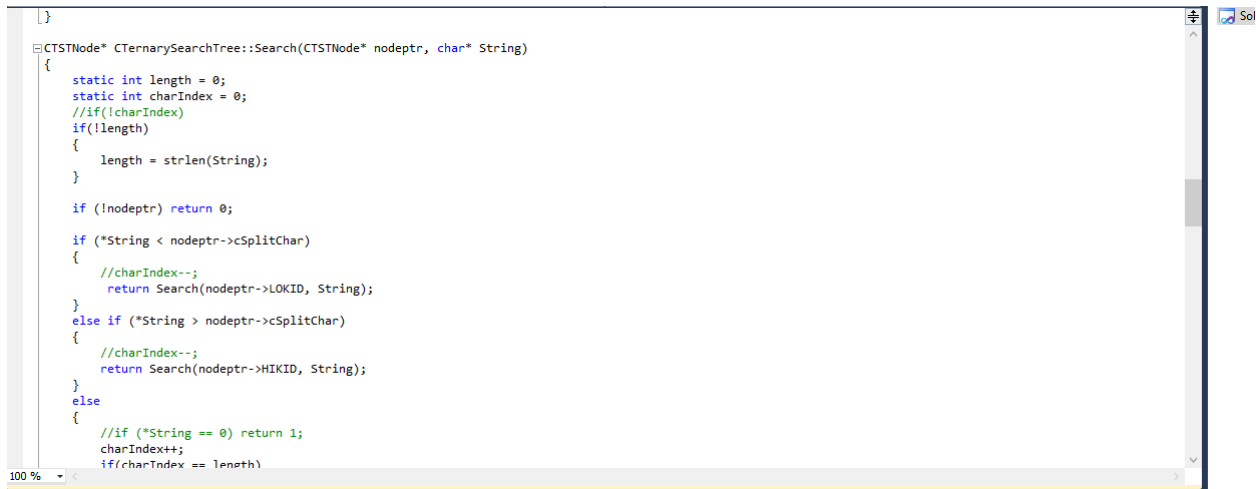
        else if (*SplitChar == nodeptr->cSplitChar)
        {
            nodeptr->EQKID = Insert(nodeptr->EQKID, ++SplitChar, meaning);
        }
    }
}
```

Figure 6: Code of task 2

(Source: Created using Microsoft Visual Studio)

From the entire dictionary code, this code helps the user to find the word by using a character of the word. In order to testing whether this part is executing in a proper manner or not the researcher has given “zzz” in the finding box and press the ok button. After that, all the word which contains zzz, are visualized in the computer or mobile screen. For executing this code in a proper manner the researcher has to execute some condition in a serial manner and fire this job the researcher has used the if-else statement. In the C++ programming language, the if-else

statement is used to holds the conditions, in case the condition under the if statement becomes true, the system visualized the if statement and in case the condition under the if the statement becomes false then the else statement became visualization the screen (Delgado-Pérez *et al.* 2017, p. e1630).



```
CTSTNode* CTernarySearchTree::Search(CTSTNode* nodeptr, char* String)
{
    static int length = 0;
    static int charIndex = 0;
    //if(!charIndex)
    if(!length)
    {
        length = strlen(String);
    }

    if (!nodeptr) return 0;

    if (*String < nodeptr->cSplitChar)
    {
        //charIndex--;
        return Search(nodeptr->LOKID, String);
    }
    else if (*String > nodeptr->cSplitChar)
    {
        //charIndex++;
        return Search(nodeptr->HIKID, String);
    }
    else
    {
        //if (*String == 0) return 1;
        charIndex++;
        if(charIndex == length)
    }
}
```

Figure 7: Code of task 3

(Source: Created using Microsoft Visual Studio)

Figure number 7 consists of the code of task 3. This code allows the user to find the required data by using different conditions. During the execution of this code, the researcher has tried to search those word which contains q abut do not contains U, and the researcher has fin his or her required data from this code. But if the dictionary cannot find the data matched with the user requirements then it visualized error by using a popup message box.

Part 2

Basic task


```

void CTernarySearchTree::Traverse_And_Match(CTSTNode* nodeptr, char* String )
{
    if (!nodeptr) return;
    Traverse_And_Match(nodeptr->LOKID,String );
    if (nodeptr->cSplitChar)
    {
        Traverse_And_Match(nodeptr->EQKID,String);
    }

    //Som++ 10th December 2005 testing with my own version of find_sub_str
    //if((nodeptr->originalstring) && (CString(nodeptr->originalstring).Find(String) == 0))
    if((nodeptr->originalstring) && (Find_Sub_Str(String, nodeptr->originalstring) == 0))
    {
        //AfxMessageBox(nodeptr->originalstring);
        //strList.RemoveAll();
        strList.AddHead(CString(nodeptr->originalstring));
    }

    Traverse_And_Match(nodeptr->HIKID,String);
}

```

Figure 8: Task 1 Code

(Source: Created using Microsoft Visual Studio)

The upper mentioned code allows the user to find the verb and noun from the dictionary. The researcher has added this part in the dictionary code, during the dictionary creation, which helps the user to find all the noun of the dictionary and all the verb of the dictionary. In order to check this code, the researcher has used the Mango word, which is an example of a noun. In order to check the existence of the verb, the researcher gives the “do” word in the finding box. In order to check parts of speech of a word, the researcher has used some condition under the if-else statement.

```

for(int i = 0; i<length_str;)
{
    int count_of_matched = 0;
    for(int j = 0; j<length_substr;)
    {
        if(str[i] == substr[j])
        {
            if( !matched_position_updated && !matched_position)
            {
                matched_position = i;
                matched_position_updated = true;
            }

            i++;
            j++;
            matched = true;
            count_of_matched++;
        }
        else
        {
            i++;
            matched = false;
            matched_position = 0;
            matched_position_updated = false;
            break;
        }
    }
    if(matched && (count_of_matched == length_substr))
        return matched_position;
}

```

Figure 9: Task 2 Code

(Source: Created using Microsoft Visual Studio)

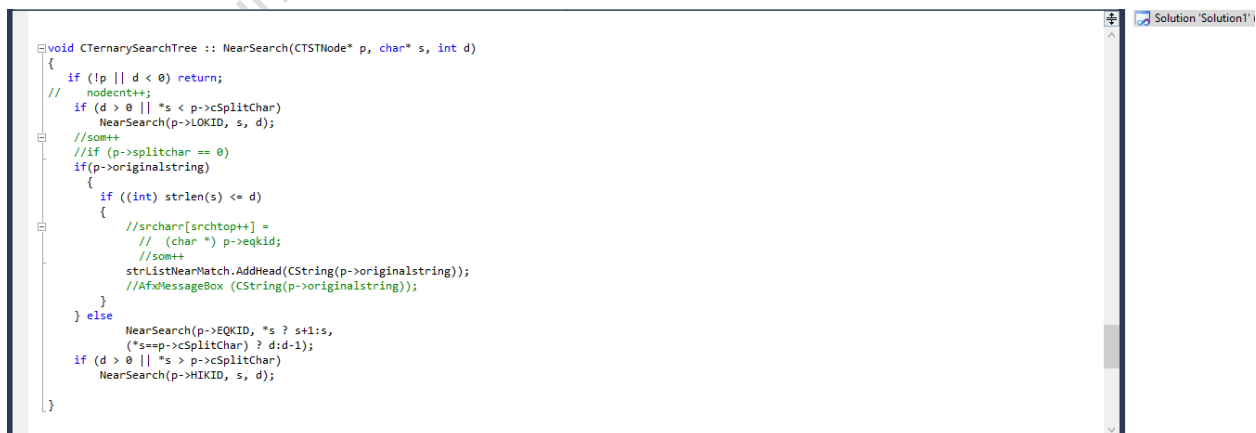
In task 2 the researcher has tried to check whether a word is a palindrome or not. In order to do this job, the researcher has used for loop and under the for loop the researcher has used if-else statement. At the first for loop, the researcher has used the length of the words as a condition and then in the second for loop the researcher has used the character of the words. Under the for loop, the if-else statement is used to check different conditions. In order to check whether a word is a palindrome or not the user to give the word in the palindrome checking box, and then the user has to press the ok button. If the word becomes palindrome then this code allows the system to visualized the “True” message on the screen but in case the word become non-palindrome, ten the system visualized “False” Massage on the screen. For the upper mentioned code, the user can check one word at a time, that is it palindrome or not.

In the computer science world, for loop is a control flow statement that is used to specify the interaction that allows code to be executed in a repeated manner (Huang *et al.* 2018, p. 20). Based on the for loop, the system first executes a condition (greater than, less than) between the given numbers, after that it did the required operation and then increment or decrement the number, and send the number for the comparison.

Syntax of the For loop: for (*statement 1*; *statement 2*; *statement 3*) {

```
// code block to be executed  
  
}
```

Intermediate tasks



```
void CTernarySearchTree :: NearSearch(CTSTNode* p, char* s, int d)  
{  
    if (!p || d < 0) return;  
    // nodecnt++;  
    if (d > 0 || *s < p->cSplitChar)  
        NearSearch(p->LOKID, s, d);  
    //som++  
    //if (p->splitchar == 0)  
    if(p->originalstring)  
    {  
        if ((int) strlen(s) <= d)  
        {  
            //srchar[srctop++] =  
            // (char *) p->eqkid;  
            //som++  
            strListNearMatch.AddHead(CString(p->originalstring));  
            //afxMessageBox (CString(p->originalstring));  
        }  
    } else  
        NearSearch(p->EQKID, *s ? s+1:s,  
        (*s==p->cSplitChar) ? d:d-1);  
    if (d > 0 || *s > p->cSplitChar)  
        NearSearch(p->HIKID, s, d);  
}
```

Figure 10: Code of task 1

(Source: Created using Microsoft Visual Studio)

By using this code the researcher has tried to find the anagrams of a word. For doing the anagrams programming in a proper manner the researcher has used the if-else statement. During the execution of this function, the researcher has given the “care” word in the anagram finding box. After that, by pressing a button the researcher has found the anagram of the care, which is “race”. From the upper mentioned image it is visualized some if-else statement are used in this case, which holds several conditions of the anagrams. In C++ if-else statement used two different codes based on the status of a condition, whether it is true or false (Zhang et al. 2017, p. 295).



```
▶ The remaining components.
import time # Import time module.

just_len = 130

print("guess a word game".upper().center(just_len, '-'))

player_name = input("\nWhat's your name?\n")

print(f"\nType 's' to stop the game at any time.\n{'#' * just_len}")

count = 0

t1 = time.time() # Notes down the time in epoch.
for tup in my_words:
    print("\nJumbled Letters: {} \nPart of speech: {} \nMeaning: {}".format(shuffler(tup[0]).upper(),
                                                                              tup[1].upper(),
                                                                              tup[2].upper()))

player_guess = input(f"Guess the word:\n").lower()
if player_guess == 's':
    break
elif player_guess == tup[0]:
    print(f"\nCORRECT!\n{'#' * just_len}")
    count += 1
```

Figure 11: Code of task 2

(Source: Created using Microsoft Visual Studio)

In task 2, the researcher has tried to create a guessing game by using python programming. For playing this game the user has to give his or her name in the main filed. After that, the system displays a definition and length of the noun on the screen as a clue of the guessing word and based on this clue the user has to guesses the correct word. For finding each word the user gets 3 chances, in the first chance, if the user cannot find the word then one character of the word

become reveal, in second-chance, if the user cannot find the word again then another one character become disclosed, but if in second chance the user failed then he or she do not get any points. After completing the game the user can visualize their total point on the screen. This game helps the user to refresh the mind and gather more word knowledge. It is one type of word knowledge testing game.

For creating this the researcher has to use different conditions, statement and commands of the python language, such as print, for loop, if-else statement and so on. The print statement is used to visualize the Messages on the screen, where for loop is used to execute the same conditions several times (Springer and Masuhara, 2018, p. 6).

Conclusion

This part is the ending part of the project and in this part; the researcher is going to conclude this project in an effective manner. In this part, all the information about the entire project is summarized by the researcher. This project is created on object-oriented programming with C++. During this project, the researcher has to develop different C++ codes in order to solve some problems and these codes help the researcher to achieve all the requirements of the project. In this project, an English dictionary is properly created by the researcher and different tasks, which are based on the dictionary, also solved by the researcher. From this project, a learner can learn the application of the C++ programs in a clear manner.

References

- Alfianto, E., Rusydi, F., Aisyah, N.D., Fadilla, R.N., Dipojono, H.K. and Martoprawiro, M.A., 2017, May. Implementation of density functional theory method on object-oriented programming (C++) to calculate energy band structure using the projector augmented wave (PAW). In *Journal of Physics: Conference Series* (Vol. 853, No. 1, p. 012043). IOP Publishing.
- Delgado-Pérez, P., Segura, S. and Medina-Bulo, I., 2017. Assessment of C++ object-oriented mutation operators: A selective mutation approach. *Software Testing, Verification and Reliability*, 27(4-5), p.e1630.
- Huang, W., Absil, P.A., Gallivan, K.A. and Hand, P., 2018. ROPTLIB: an object-oriented C++ library for optimization on Riemannian manifolds. *ACM Transactions on Mathematical Software (TOMS)*, 44(4), pp.1-21.
- Springer, M. and Masuhara, H., 2018, February. Ikra-Cpp: A C++/CUDA DSL for object-oriented programming with structure-of-arrays layout. In *Proceedings of the 2018 4th Workshop on Programming Models for SIMD/Vector Processing* (pp. 1-9).
- Zhang, T., Ma, R., Huang, X. and Yao, L., 2017, September. based Course Teaching of " C++ Programming" Combining the FKM Teaching Method. In *2017 3rd International Conference on Social Science and Higher Education* (pp. 295-298). Atlantis Press.