UTM UNIVERSITI TEKNOLOGI MALAYSIA

School of Professional and Continuing Education (SPACE)

ASSIGNMENT 2

SEARCHING AND SORTING

DDWD 2733

SECTION 40

Group Members:

1)	Ammar Danish bin Anwar Zamsani	A21DW0116
2)	Azry Fikri Iskandar bin Roslan	A21DW0997
3)	Bahrul Ilmi bin Mohd Yuswandy	A21DW0945
4)	Ikhwan Hafizi bin Mohamad Jahari	A21DW1006

Submission Date	:	31st October 2022
Course	:	2 DDWD Diploma in Computer Science
Lecturer’s Name	:	Madam Rokiah binti Bahari

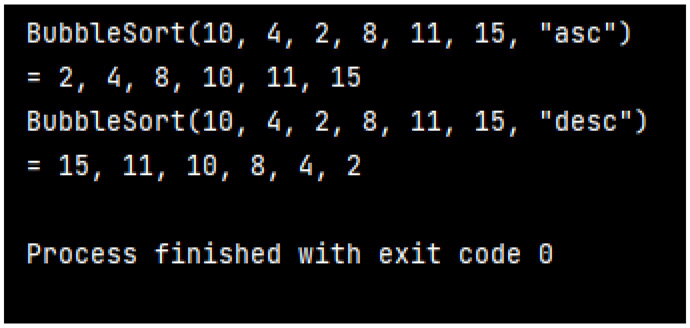
QUESTION 1 : BUBBLE SORT ALGORITHM

CODE:

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> BubbleSort(vector<int>, string);
vector<int> BubbleSort(vector<int> arr, string order){
   int n = arr.size();
   for(int i=0; i<n; i++){
       for(int j=0; j<n-i-1; j++){
           if(order == "asc"){
               if (arr[j] > arr[j + 1]) {
                   //swap these elements
                   arr[j] = arr[j] + arr[j + 1];
                   arr[j + 1] = arr[j] - arr[j + 1];
                   arr[j] = arr[j] - arr[j + 1];
               }
           }
           else if(order == "desc"){
               if (arr[j] < arr[j + 1]) {
                   //swap these elements
                   arr[j] = arr[j] + arr[j + 1];
                   arr[j + 1] = arr[j] - arr[j + 1];
                   arr[j] = arr[j] - arr[j + 1];
               }
           }
       }
   }
   return arr;
}

int main() {
   vector<int> array = {10, 4, 2, 8, 11, 15};

   vector<int> arrayAsc = BubbleSort(array, "asc");
   vector<int> arrayDesc = BubbleSort(array, "desc");
   int n = array.size();

   cout << "BubbleSort(";
   for(int i=0; i<n; i++){
       cout << array[i] << ((i<n-1) ? ", " :  "");
   }
   cout << ", \"asc\") \n= ";
   for(int i=0; i<n; i++){
       cout << arrayAsc[i] << ((i<n-1) ? ", " :  "");
   }
   cout << endl;

   cout << "BubbleSort(";
   for(int i=0; i<n; i++){
       cout << array[i] << ((i<n-1) ? ", " :  "");
```

```
    }
    cout << ", \"desc\") \n= ";
    for(int i=0; i<n; i++){
        cout << arrayDesc[i] << ((i<n-1) ? ", " :  "");
    }
    cout << endl;
    return 0;
}
```

OUTPUT:



```
BubbleSort(10, 4, 2, 8, 11, 15, "asc")
= 2, 4, 8, 10, 11, 15
BubbleSort(10, 4, 2, 8, 11, 15, "desc")
= 15, 11, 10, 8, 4, 2

Process finished with exit code 0
```

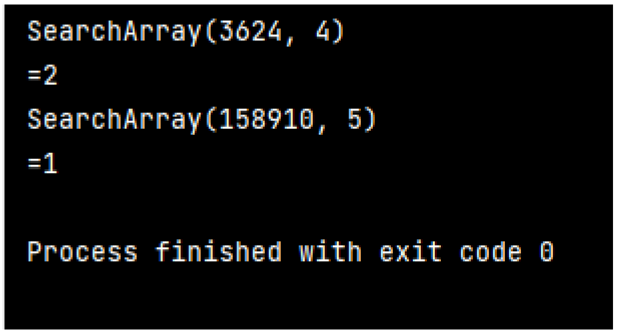
QUESTION 2 : BINARY SEARCH

CODE:

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> BubbleSort(vector<int>, string);
int BinarySearch(vector<int>, int);
vector<int> slice(vector<int>, int, int);

vector<int> BubbleSort(vector<int> arr, string order){
   int n = arr.size();
   for(int i=0; i<n; i++){
       for(int j=0; j<n-i-1; j++){
           if(order == "asc"){
               if (arr[j] > arr[j + 1]) {
                   //swap these elements
                   arr[j] = arr[j] + arr[j + 1];
                   arr[j + 1] = arr[j] - arr[j + 1];
                   arr[j] = arr[j] - arr[j + 1];
               }
           }
           else if(order == "desc"){
               if (arr[j] < arr[j + 1]) {
                   //swap these elements
                   arr[j] = arr[j] + arr[j + 1];
                   arr[j + 1] = arr[j] - arr[j + 1];
                   arr[j] = arr[j] - arr[j + 1];
               }
           }
       }
   }
   return arr;
}

int BinarySearch(vector<int> arr, int search){
   bool sorted = true;
   int n = arr.size();
   for(int i=0; i<n-1; i++){
       if(arr[i] > arr[i+1]){
           sorted = false;
       }
   }

   if(!sorted){
       arr = BubbleSort(arr, "asc");
```

```
    }

    int first, mid, last, foundIndex=0;
    first = 0;
    mid = (int)((arr.size()-1)/2);
    last = arr.size()-1;

    if(arr[mid] < search){
        vector<int> slicedVector = slice(arr, mid+1, last);
        foundIndex = mid+1 + BinarySearch(slicedVector, search);
    }
    else if(arr[mid] > search){
        vector<int> slicedVector = slice(arr, first, mid-1);
        foundIndex = BinarySearch(slicedVector, search);
    }
    else if(arr[mid] == search){
        foundIndex = mid;
    }

    return foundIndex;
}

vector<int> slice(vector<int> arr, int x, int y){
    vector<int> sliced;

    for(int i=x; i<=y; i++){
        sliced.insert(sliced.end(), arr[i]);
    }

    return sliced;
}

int main(){
    vector<int> array1 = {3, 6, 2, 4};
    vector<int> array2 = {1, 5, 8, 9, 10};

    cout << "SearchArray(";
    for(int i=0; i<array1.size(); i++){
        cout << array1[i];
    }
    cout << ", 4) \n=" << BinarySearch(array1, 4) << endl;

    cout << "SearchArray(";
    for(int i=0; i<array2.size(); i++){
        cout << array2[i];
    }
    cout << ", 5) \n=" << BinarySearch(array2, 5) << endl;
}
```

OUTPUT:



```
SearchArray(3624, 4)
=2
SearchArray(158910, 5)
=1

Process finished with exit code 0
```

QUESTION 3 : STRING

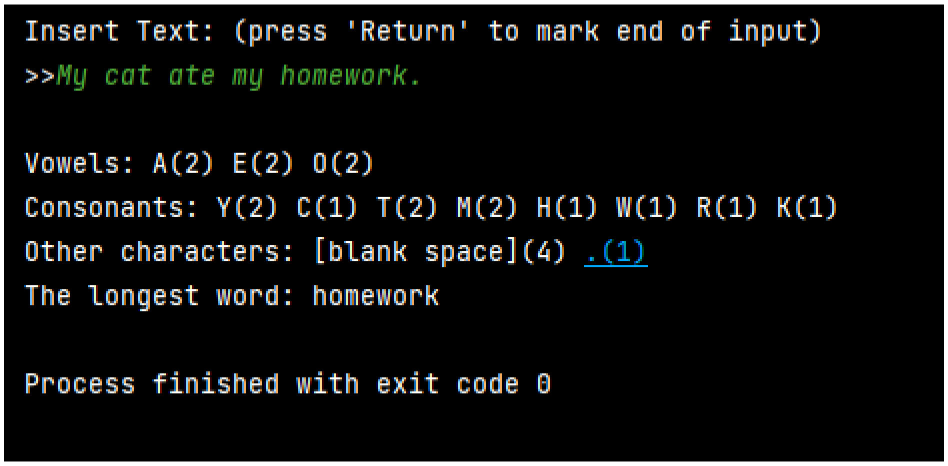
CODE:

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

vector<int> BubbleSort(vector<int>, string);
vector<int> BubbleSort(vector<int> arr, string order){
   int n = arr.size();
   for(int i=0; i<n; i++){
       for(int j=0; j<n-i-1; j++){
           if(order == "asc"){
               if (arr[j] > arr[j + 1]) {
                   //swap these elements
                   arr[j] = arr[j] + arr[j + 1];
                   arr[j + 1] = arr[j] - arr[j + 1];
                   arr[j] = arr[j] - arr[j + 1];
               }
           }
           else if(order == "desc"){
               if (arr[j] < arr[j + 1]) {
                   //swap these elements
                   arr[j] = arr[j] + arr[j + 1];
                   arr[j + 1] = arr[j] - arr[j + 1];
                   arr[j] = arr[j] - arr[j + 1];
               }
           }
       }
   }
   return arr;
}

struct Node{
   char data;
   int frequency = 1;
   Node* link;
};

class LinkedList{
   private:
       int count;
       Node* pHead;
       Node* pPre;
   public:
```

```
LinkedList(){
    count = 0;
    pHead = nullptr;
    pPre = nullptr;
}

void add(char data){
    if(search(data)){
        pPre->frequency++;
    }
    else{
        Node *pNew = new Node;
        pNew->data = data;
        pNew->link = nullptr;

        if (pHead == nullptr) {
            pHead = pNew;
        } else {
            pPre->link = pNew;
        }
        count++;
    }
}

void addWithFrequency(char data, int freq){
    for(int i=1; i<=freq; i++){
        add(data);
    }
}

bool search(char target){
    Node* pLoc = pHead;
    while(pLoc != nullptr){
        pPre = pLoc;
        if(pPre->data == target){
            return true;
        }
        pLoc = pLoc->link;
    }
    return false;
}

void print(){
    Node* pWalk = pHead;
    while(pWalk != nullptr){
        if(pWalk->data == ' '){
            cout << "[blank space](" << pWalk->frequency << ") ";
```

```cpp
            }
            else {
                cout << pWalk->data << "(" << pWalk->frequency << ") ";
            }
            pWalk = pWalk->link;
        }
    }

    Node* get(int index){
        if(index >= count)
            return nullptr;
        Node* pLoc = pHead;
        for(int i=0; i<=index-1; i++){
            pLoc = pLoc->link;
        }

        return pLoc;
    }

    int getCount(){
        return count;
    }
};

bool isVowel(char data){
    char v[5] = {'A', 'E', 'I', 'O', 'U'};

    for(int i=0; i<5; i++){
        if(data == v[i]) return true;
    }
    return false;
}

bool isAlpha(char data){
    if(data >= 'A' && data <= 'Z'){
        return true;
    }
    else{
        return false;
    }
}

int main(){
    string sentence;

    cout << "Insert Text: (press 'Return' to mark end of input)\n>>";
    cin.ignore();
    getline(cin, sentence);
    LinkedList pairList;
    for(int i=0; i<sentence.size(); i++){
        pairList.add((char)toupper(sentence.at(i)));
    }

    LinkedList vowels, consonants, specialChar;
    for(int i=0; i<pairList.getCount(); i++){
        Node* current = pairList.get(i);
        if(isAlpha(current->data) && isVowel(current->data)){
            vowels.addWithFrequency(current->data, current->frequency);
        }
        else if(isAlpha(current->data) && !isVowel(current->data)){
            consonants.addWithFrequency(current->data, current->frequency);
        }
        else if(!isAlpha(current->data)){
            specialChar.addWithFrequency(current->data, current->frequency);
        }
    }

    string count="", longest="";
    string sentenceAppended = sentence + " ";
    for(int i=0; i<sentenceAppended.size(); i++){
        if(!isAlpha(toupper(sentenceAppended.at(i)))){
            if (longest.size() < count.size()) {
                longest = count;
            }
            count = "";
        }
        else{
            count += sentenceAppended.at(i);
        }
    }

    cout << endl;
    cout << "Vowels: ";
    vowels.print();
    cout << endl;
    cout << "Consonants: ";
    consonants.print();
    cout << endl;
    cout << "Other characters: ";
    specialChar.print();

    cout << "\nThe longest word: " << longest << endl;

}
```

OUTPUT



```
Insert Text: (press 'Return' to mark end of input)
>>My cat ate my homework.

Vowels: A(2) E(2) O(2)
Consonants: Y(2) C(1) T(2) M(2) H(1) W(1) R(1) K(1)
Other characters: [blank space](4) .(1)
The longest word: homework

Process finished with exit code 0
```