Alexandria University
Faculty of Engineering
Computer and Systems Engineering Department
CSE 214: Discrete Structures

Lab 2 – Power set

Team Members

1. ID: 20011574 – محمد رفيق محمد السيد ابراهيم
2. ID: 20011923 – مصطفى خالد كمال احمد زايد

First: Problem Statement

Given a set represented as an array list of distinct strings, you have to generate all possible subsets from the set.

The problem should be solved twice, using an iterative approach and a recursive approach.

Second: Used Data Structures

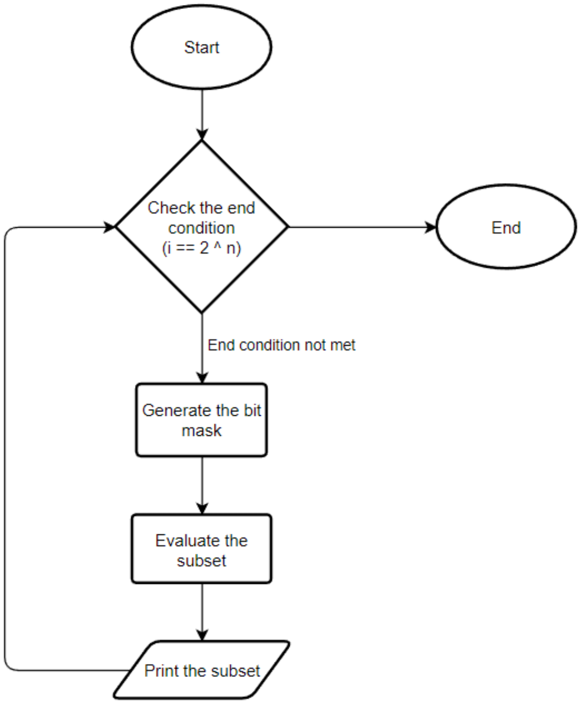
Only basic arrays are used to implement the program:

`string ans[n];` → holds the result of each iteration during the iterative approach.

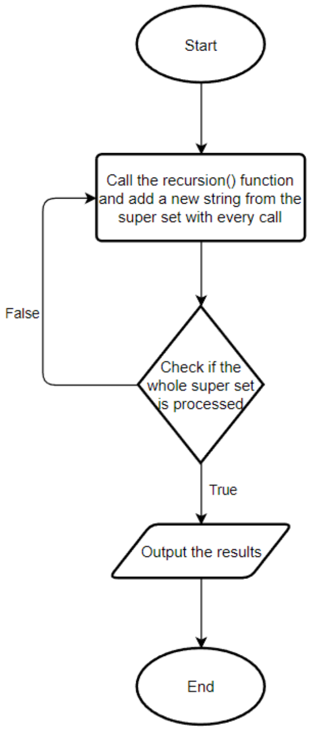
`string input_array[n];` → holds the power set.

Third: Flow Charts

Iterative Approach

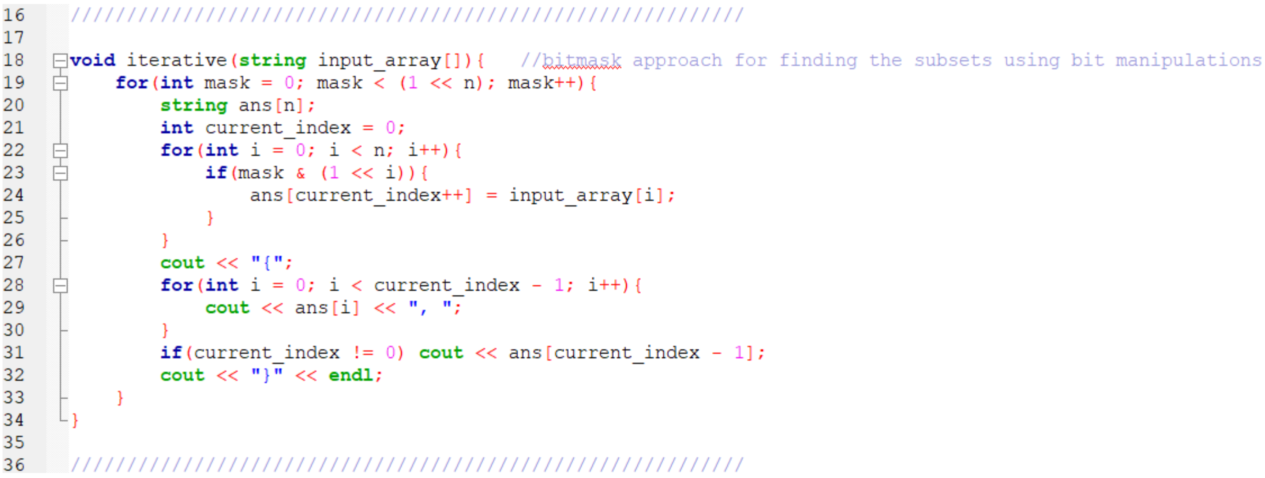


Recursive Approach


Start
Call the recursion() function and add a new string from the super set with every call
False
Check if the whole super set is processed
True
Output the results
End

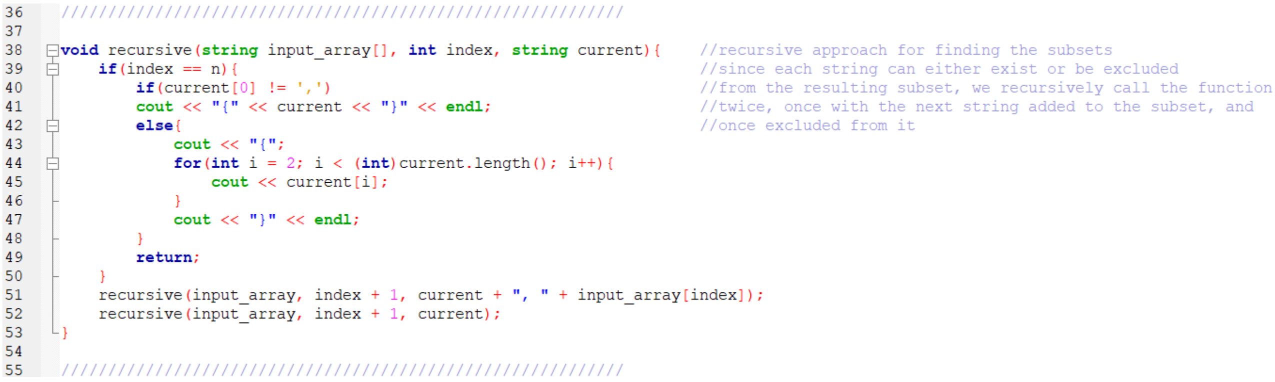
Fourth: Code Snippets

Iterative:



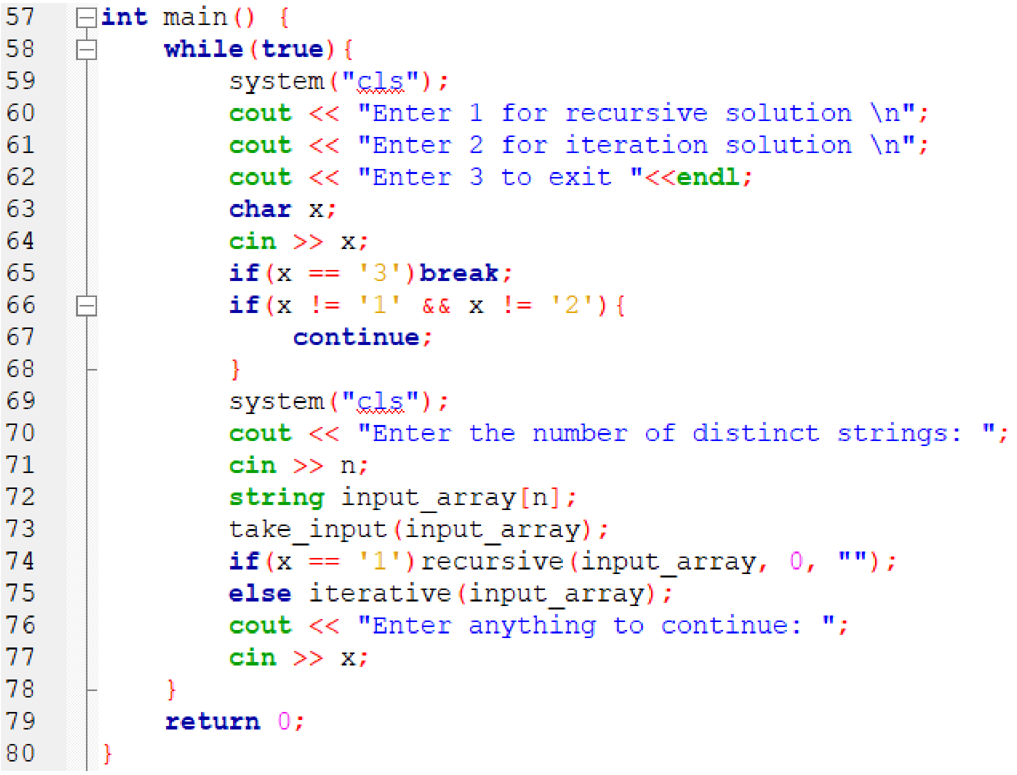
```
////////////////////////////////////////////////////////////

void iterative(string input_array[]){    //bitmask approach for finding the subsets using bit manipulations
    for(int mask = 0; mask < (1 << n); mask++){
        string ans[n];
        int current_index = 0;
        for(int i = 0; i < n; i++){
            if(mask & (1 << i)){
                ans[current_index++] = input_array[i];
            }
        }
        cout << "{";
        for(int i = 0; i < current_index - 1; i++){
            cout << ans[i] << ", ";
        }
        if(current_index != 0) cout << ans[current_index - 1];
        cout << "}" << endl;
    }
}

////////////////////////////////////////////////////////////
```

Recursive:



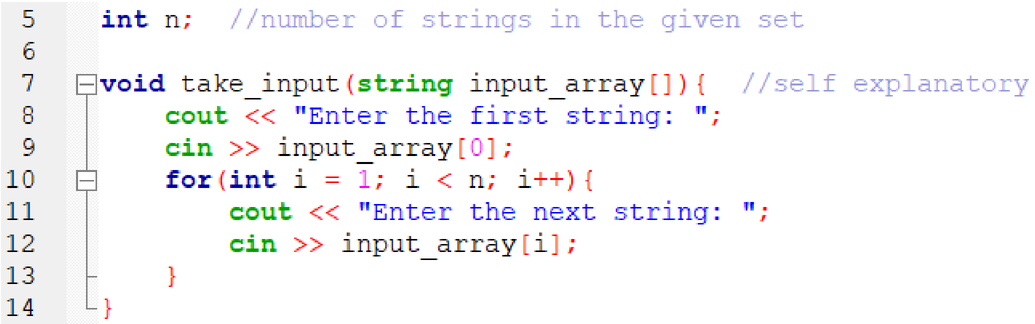
```
////////////////////////////////////////////////////////////

void recursive(string input_array[], int index, string current){    //recursive approach for finding the subsets
    if(index == n){                                                  //since each string can either exist or be excluded
        if(current[0] != ',')                                        //from the resulting subset, we recursively call the function
        cout << "{" << current << "}" << endl;                       //twice, once with the next string added to the subset, and
        else{                                                        //once excluded from it
            cout << "{";
            for(int i = 2; i < (int)current.length(); i++){
                cout << current[i];
            }
            cout << "}" << endl;
        }
        return;
    }
    recursive(input_array, index + 1, current + ", " + input_array[index]);
    recursive(input_array, index + 1, current);
}

////////////////////////////////////////////////////////////
```

Main:



```cpp
int main()  {
    while(true){
        system("cls");
        cout << "Enter 1 for recursive solution \n";
        cout << "Enter 2 for iteration solution \n";
        cout << "Enter 3 to exit "<<endl;
        char x;
        cin >> x;
        if(x == '3')break;
        if(x != '1' && x != '2'){
            continue;
        }
        system("cls");
        cout << "Enter the number of distinct strings: ";
        cin >> n;
        string input_array[n];
        take_input(input_array);
        if(x == '1')recursive(input_array, 0, "");
        else iterative(input_array);
        cout << "Enter anything to continue: ";
        cin >> x;
    }
    return 0;
}
```

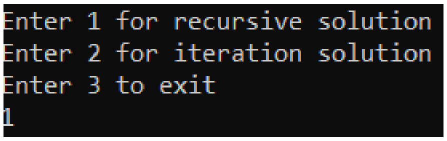
Taking input:



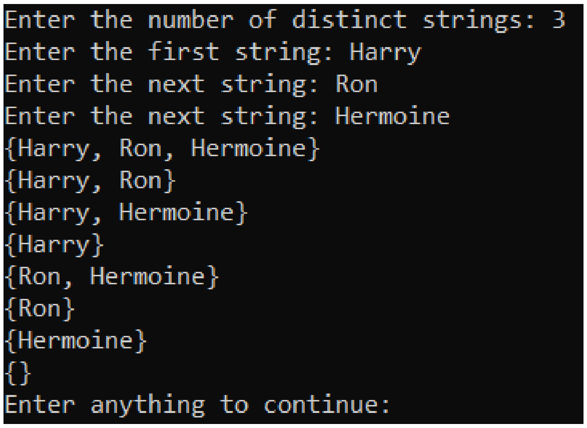
```cpp
int n;  //number of strings in the given set

void take_input(string input_array[]){  //self explanatory
    cout << "Enter the first string: ";
    cin >> input_array[0];
    for(int i = 1; i < n; i++){
        cout << "Enter the next string: ";
        cin >> input_array[i];
    }
}
```

Fifth: Sample runs

Recursive:

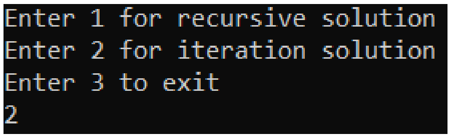


```
Enter 1 for recursive solution
Enter 2 for iteration solution
Enter 3 to exit
1
```

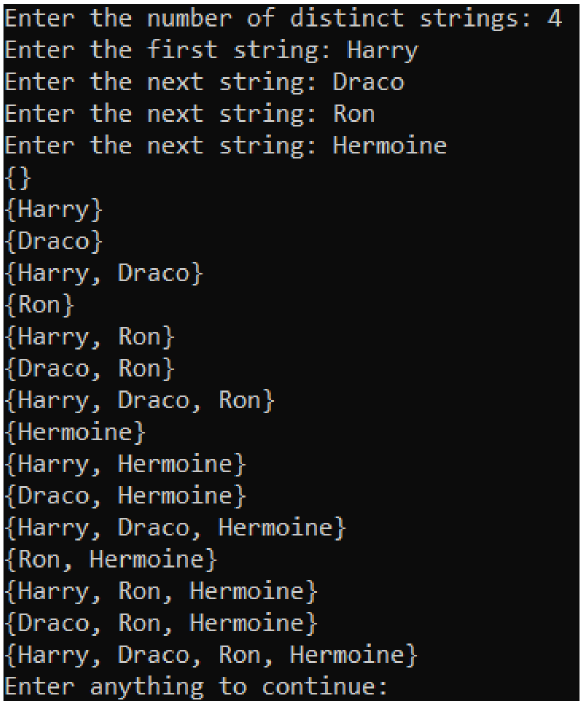


```
Enter the number of distinct strings: 3
Enter the first string: Harry
Enter the next string: Ron
Enter the next string: Hermoine
{Harry, Ron, Hermoine}
{Harry, Ron}
{Harry, Hermoine}
{Harry}
{Ron, Hermoine}
{Ron}
{Hermoine}
{}
Enter anything to continue:
```

Iterative:



```
Enter 1 for recursive solution
Enter 2 for iteration solution
Enter 3 to exit
2
```



```
Enter the number of distinct strings: 4
Enter the first string: Harry
Enter the next string: Draco
Enter the next string: Ron
Enter the next string: Hermoine
{}
{Harry}
{Draco}
{Harry, Draco}
{Ron}
{Harry, Ron}
{Draco, Ron}
{Harry, Draco, Ron}
{Hermoine}
{Harry, Hermoine}
{Draco, Hermoine}
{Harry, Draco, Hermoine}
{Ron, Hermoine}
{Harry, Ron, Hermoine}
{Draco, Ron, Hermoine}
{Harry, Draco, Ron, Hermoine}
Enter anything to continue:
```

Sixth: Important Assumptions and Details

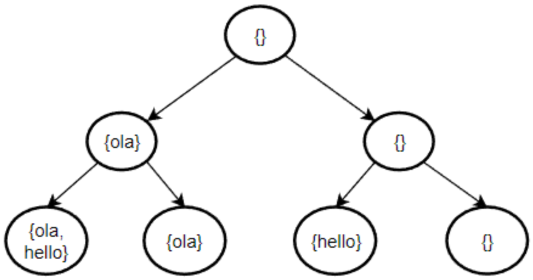
Important code details are commented in the code snippets above.

We did cover most of the wrong input cases, however, some minor corner cases were not covered assuming the input correctness by the user.

The iterative method was done using an algorithm that takes the number of strings int the power set and calculates the total number of subsets, then considers the binary representation of a created “mask” that we use to find the elements of the power set that will be included in that specific subset.

The recursive method was done by assuming that each string in the power set can either be included or excluded from the current subset, therefore by calling the recursive function twice (once including it, and once excluding it) we can generate all possible combinations of the elements in the super set. The following diagram is a simple example:

Consider a power set containing 2 strings: ola, hello



<- These are the resulting subsets