Assignment

Lab #3

Course

CIS-7

Section

27168

Due Date

September 19th, 2021

Author

Matthew Mickey

Github repo

https://github.com/PocketPiggy/MVC_Fall_CIS-7/tree/master/Labs/Lab4-Permutations

1. Using the above example 1 of next_permutation function, create a program in C++ that includes an array of 4 elements and uses the next_permutation function.

```
ling@pocketpiggy:~/Code/school/CIS_7/Labs/Lab_4/Lab_4-Permutations
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  g++ main.cpp -o Lab_4                    05:47:44
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  ./Lab_4                                 05:48:43
Problem 1

The 4! possible permutations with 4 elements:
1: 3 4 13 14
2: 3 4 14 13
3: 3 13 4 14
4: 3 13 14 4
5: 3 14 4 13
6: 3 14 13 4
7: 4 3 13 14
8: 4 3 14 13
9: 4 13 3 14
10: 4 13 14 3
11: 4 14 3 13
12: 4 14 13 3
13: 13 3 4 14
14: 13 3 14 4
15: 13 4 3 14
16: 13 4 14 3
17: 13 14 3 4
18: 13 14 4 3
19: 14 3 4 13
20: 14 3 13 4
21: 14 4 3 13
22: 14 4 13 3
23: 14 13 3 4
24: 14 13 4 3

After loop: 3 4 13 14

1a. Calculate the variation of arrangment in Example 1 program using N!: 4! = 4 * 3 * 2 * 1 = 24
1b. The arrangment of the elements are in ascending order.
1c. Replacing the values yields the same process. We also still have 24 total permutations as the previous example.
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations                                          05:48:46
```

```cpp
void prob1() {
    int i = 1, len = 4;
    int myints[] = {13, 14, 3, 4,};
    //  Sort array, ascending
    sort(myints, myints + 4);

    cout << "The 4! possible permutations with 4 elements: " << endl;
    do {
        cout << i << ": " << myints[0] << " " << myints[1] << " " << myints[2]
             << " " << myints[3] << endl;
        i++;
        //  Stops when array is in descending order
    } while (next_permutation(myints, myints + 4));

    cout << endl;

    cout << "After loop: ";
    for (int i = 0; i < len; i++) {
        cout << myints[i] << " ";
    }
    cout << endl << endl;

    cout << "1a. Calculate the variation of arrangment in Example 1 program using N!:"
         << " 4! = 4 * 3 * 2 * 1 = 24";
    cout << endl;
    cout << "1b. The arrangment of the elements are in ascending order." << endl;
    cout << "1c. Replacing the values yields the same process. We also still have 24"
         << " total permutations as the previous example.";

    cout << endl;
}
```

2. Refer to the above example 2 and create a C++ program that contains prev_permutation function and an array of 5 elements.

```
ling@pocketpiggy:~/Code/school/CIS_7/Labs/Lab_4/Lab_4-Permutations
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  g++ main.cpp -o Lab_4                    05:50:47
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  ./Lab_4                                 05:50:51
Problem 2

The 4! possible permutations with 3 elements:

1: 20 19 18 4
2: 20 19 4 18
3: 20 18 19 4
4: 20 18 4 19
5: 20 4 19 18
6: 20 4 18 19
7: 19 20 18 4
8: 19 20 4 18
9: 19 18 20 4
10: 19 18 4 20
11: 19 4 20 18
12: 19 4 18 20
13: 18 20 19 4
14: 18 20 4 19
15: 18 19 20 4
16: 18 19 4 20
17: 18 4 20 19
18: 18 4 19 20
19: 4 20 19 18
20: 4 20 18 19
21: 4 19 20 18
22: 4 19 18 20
23: 4 18 20 19
24: 4 18 19 20

After loop: 20 19 18 4

2a. Calculate the possible arrangement of numbers that the Example 2 program will display:  N = 4, 4! = 24
2b. Opposite of the previous example; descending.
2c. Similar outcome. It doesn't matter what the actual value is if they're higher. It's still going off of the position
 of the elements.
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations                                          05:50:53
```

```cpp
void prob2() {
    int i = 1, len = 4;
    //int myints[] = {1, 2, 3, 4,};
    int myints[] = {18, 19, 20, 4,};
    sort(myints, myints + 4);
    reverse(myints, myints + 4);

    cout << "The 4! possible permutations with 3 elements: " << endl << endl;

    do {
        cout << i << ": " << myints[0] << " " << myints[1] << " " << myints[2]
             << " " << myints[3] << endl;
        i++;
    } while (prev_permutation(myints, myints + 4));
    cout << endl;

    cout << "After loop: ";
    for (int i = 0; i < len; i++) {
        cout << myints[i] << " ";
    }
    cout << endl << endl;

    cout << "2a. Calculate the possible arrangement of numbers that the Example 2 program will display: "
         << " N = 4, 4! = 24" << endl;
    cout << "2b. Opposite of the previous example; descending." << endl;
    cout << "2c. Similar outcome. It doesn't matter what the actual value is if they're higher."
         << " It's still going off of the position of the elements." << endl;
}
149
```

3. Following the above example 3 of is_permutation function, create a C++ program that contains 2 arrays of 5 elements that have the same elements in different order.

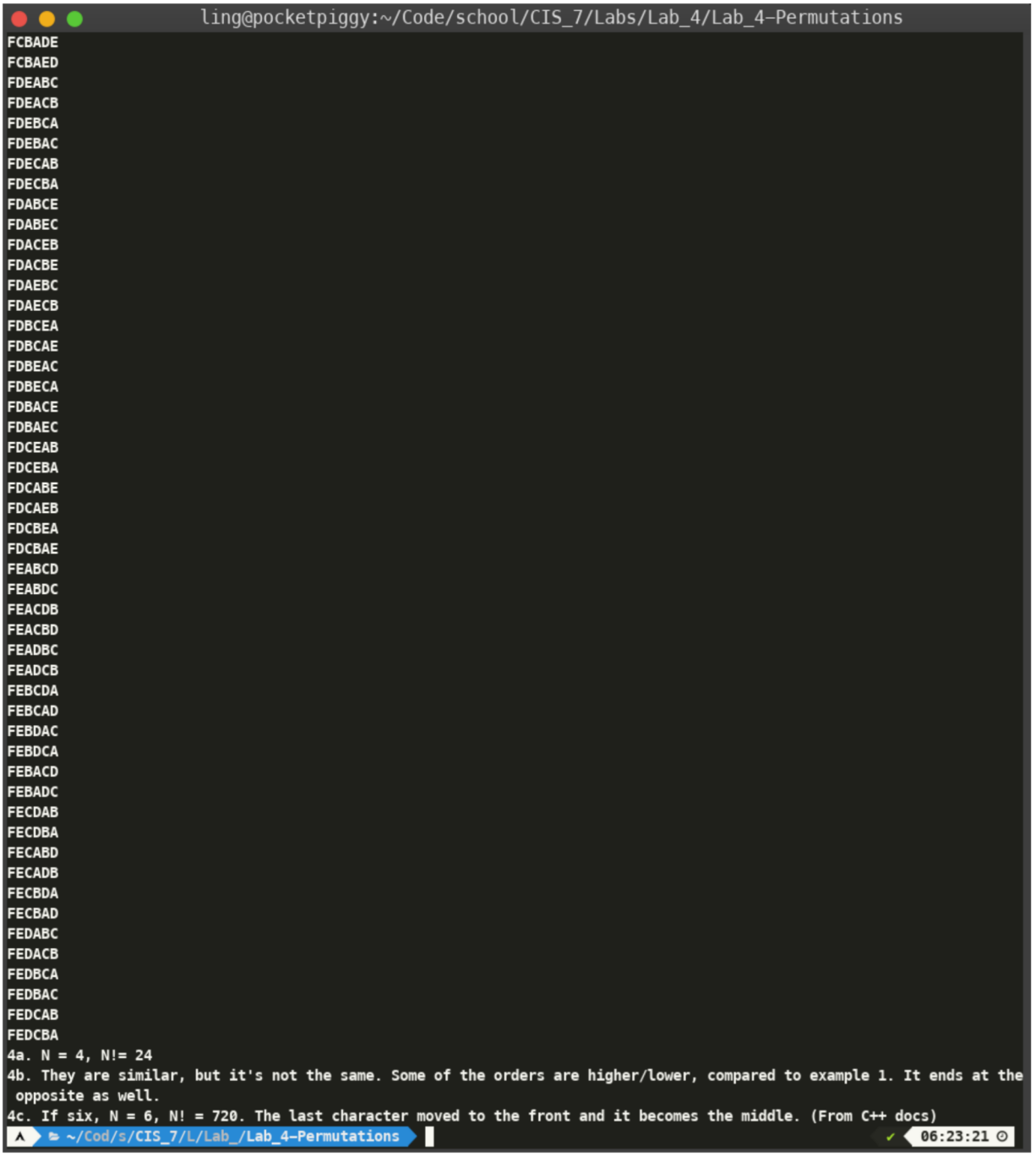
```
ling@pocketpiggy:~/Code/school/CIS_7/Labs/Lab_4/Lab_4-Permutations
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  g++ main.cpp -o Lab_4                06:04:50
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  ./Lab_4                             06:05:00
Problem 3



B is a permutation of A

3a. Calculate the number of arrangments or permutation given arr_A = {1, 5, 0, 3, 2}: N = 5, N! = 120
3b. It says that B is a permutation of A. This is because arr_B is a just a rearranged order of the original.


Now changing arr_B's elements...

B is not a permutation of A

3c. It goes to the else branch because it's false, there are different numbers so they're not permutations.


~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations                                      06:05:01
```

```cpp
void prob3() {
    cout << endl << endl;
    int arr_A[] = {1, 5, 0, 3, 2};
    //int arr_B[] = {2, 0, 5, 3, 1};
    int arr_B[] = {0, 5, 3, 1, 2};
    int arr_C[] = {7, 8, 9, 1, 2};  //  Second array for 3c

    //  Checks arr B includes all eles of arr A
    if (is_permutation(arr_A, arr_A + 5, arr_B)) {
        cout << "B is a permutation of A";
    }
    else {
        cout << "B is not a permutation of A";
    }


    cout << endl << endl;
    cout << "3a. Calculate the number of arrangments or permutation given arr_A ="
         << " {1, 5, 0, 3, 2}: "
         << "N = 5, N! = 120" << endl;
    cout << "3b. It says that B is a permutation of A. This is because arr_B is a"
         << " just a rearranged order of the original." << endl;
    cout << endl << endl;
    //  Lazy; Arr_C in place for Arr_B
    cout << "Now changing arr_B's elements..." << endl << endl;
    if (is_permutation(arr_A, arr_A + 5, arr_C)) {
        cout << "B is a permutation of A";
    }
    else {
        cout << "B is not a permutation of A";
    }
    cout << endl << endl;

    cout << "3c. It goes to the else branch because it's false, there are different"
         << " numbers so they're not permutations." << endl;
    cout << endl << endl;
}
```

4. Use the above example 4 program and create the Heap's algorithm program in C++.

```
ling@pocketpiggy:~/Code/school/CIS_7/Labs/Lab_4/Lab_4-Permutations
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  g++ main.cpp -o Lab_4                    06:21:26
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations  ./Lab_4                                 06:21:52
Problem 4

ABCD
ABDC
ACDB
ACBD
ADBC
ADCB
BCDA
BCAD
BDAC
BDCA
BACD
BADC
CDAB
CDBA
CABD
CADB
CBDA
CBAD
DABC
DACB
DBCA
DBAC
DCAB
DCBA


4a. N = 4, N!= 24
4b. They are similar, but it's not the same. Some of the orders are higher/lower, compared to example 1. It ends at the
 opposite as well.
4c. If six, N = 6, N! = 720. The last character moved to the front and it becomes the middle. (From C++ docs)
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations                                          06:21:53
```

```cpp
int main(int argc, char** argv) {
    string str0 = "ABCD";
    string str1 = "ABCDEF";
    //example1();
    //cout << "Problem 1" << endl << endl;
    //prob1();
    //cout << endl << endl;
    //cout << "Problem 2" << endl << endl;
    //example2();
    //prob2();
    //cout << endl << endl;
    //cout << "Problem 3" << endl << endl;
    //example3();
    //prob3();
    //cout << endl << endl;
    cout << "Problem 4" << endl << endl;
    example4(str0, "");
    cout << endl << endl;
    //example4(str1, "");
    prob4();
    //cout << endl << endl;
    return 0;
}

void example4(string str, string out) {
    if (str.size() == 0) {
        cout << out << endl;
    }

    for (int i = 0; i < str.size(); i++) {
        //  Remove first char and put it to out
        example4(str.substr(1), out + str[0]);  //  Permute
        //  Rotate string so second char moves to beginning
        rotate(str.begin(), str.begin() + 1, str.end());
    }
}
```


ling@pocketpiggy:~/Code/school/CIS_7/Labs/Lab_4/Lab_4-Permutations
FCBADE
FCBAED
FDEABC
FDEACB
FDEBCA
FDEBAC
FDECAB
FDECBA
FDABCE
FDABEC
FDACEB
FDACBE
FDAEBC
FDAECB
FDBCEA
FDBCAE
FDBEAC
FDBECA
FDBACE
FDBAEC
FDCEAB
FDCEBA
FDCABE
FDCAEB
FDCBEA
FDCBAE
FEABCD
FEABDC
FEACDB
FEACBD
FEADBC
FEADCB
FEBCDA
FEBCAD
FEBDAC
FEBDCA
FEBACD
FEBADC
FECDAB
FECDBA
FECABD
FECADB
FECBDA
FECBAD
FEDABC
FEDACB
FEDBCA
FEDBAC
FEDCAB
FEDCBA
4a. N = 4, N!= 24
4b. They are similar, but it's not the same. Some of the orders are higher/lower, compared to example 1. It ends at the
opposite as well.
4c. If six, N = 6, N! = 720. The last character moved to the front and it becomes the middle. (From C++ docs)
~/Cod/s/CIS_7/L/Lab_/Lab_4-Permutations
06:23:21