سلسلة الخوارزميات وحل المشاكل - المستوى الثاني

PROGRAMMING ADVICES

LEARN THE RIGHT WAY

26+ Years of Experience

Mohammed Abu-Hadhoud

MBA, PMOC, PgMP®, PMP®, PMI-RMP®, CM, ITILF, MCPD, MCSD

ProgrammingAdvices.com

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Algorithms & Problem Solving

Level 2 (Course #5)

انا الطالب فتحي مصطفى، هذا الملف جمعت فيه أغلب (الفانكشن و البروسيجر) من الكورس الخامس الخوارزميات وحل المشاكل المستوى الثاني.

في هذا الجدول تجدون أسماء كل (فانكشن) كل ما عليك هو الضغط على إسم أي فانكشن تريده وسيأخذك الملف إلى الأكواد التابعة لها مباشرةً.

أعتذر جداً عن كل خطأ إذا كان موجوداً في هذا الملف.

ادعوا لي و لأستاذ محمد أبو هدهود و لوالدينا بالخير.

تنويه: لا تستخدم هذه الأكواد الجاهز إلا في الحال قد شاهدت فيديوهات الشرح للدرس وفهمت كيف يعمل الكود بشكل صحيح لان فيها خبرات ستفتقدها في حال اهمالها ، لكي تتجنب المشاكل مستقبلاً.

```cpp
// Solution #1/2
// Multiplication Table from 1 to 10

string ColumSperator(int Counter)
{
	if (Counter < 10)
		return "   |";
	else
		return "  |";
}
```

```cpp
// Solution #2/2
// Print All Prime Numbers From 1 to N

int ReadPositiveNumber(string Message)
{
	int Number = 0;
	do
	{
		cout << Message << endl;
		cin >> Number;
	} while (Number <= 0);
	return Number;
}
```

Prime Number: لا يقبل القسمة إلا على نفسه وعلى واحد فقط.

Note: Prime number can only divide on one and on itself.

```cpp
enum enPrimeNotPrime { Prime = 1, NotPrime = 2 };

enPrimeNotPrime CheckPrime(int Number)
{
	int M = round(Number / 2);
	for (int Counter = 2; Counter <= M; Counter++)
	{
		if (Number % Counter == 0)
			return enPrimNotPrime::NotPrime;
	}
	return enPrimNotPrime::Prime;
}
```

```
// Solution #3/2
// Perfect Number

// Input:
// 28
// 12

// Output:
// 28 is perfect.
// 12 is not perfect.
```

PerfectNumber:بيساوي مجموع تقسيم جميع الأعداد التي أصغر منه، ماعدا العدد نفسه من غير باقي

```
Note: Perfect Number is = Sum(all divisors)
28 = 1 + 2 + 4 + 7 + 14
6 = 1 + 2 + 3

bool isPerfectNumber(int Number)
{
      int Sum = 0;

      for (int i = 1; i < Number; i++)
      {
            if (Number % i == 0)
                  Sum += i;
      }

      return Number == Sum;
}
```

```
// Solution #4/2
// Perfect Numbers From 1 to N

// Input:
// 500

// Output:
// 6
// 28
// 496

void PrintPerfectNumbersFrom1ToN(int Number)
{
      for (int i = 1; i <= Number; i++)
      {
            if (isPerfectNumber(i))
            {
                  cout << i << endl;
            }
      }
}
```

```
// Solution #5/2
//يطبع كل رقم من العدد بالعكس

void PrintDigitsInReversedOrder(int Number)

{
	int Remainder = 0;

	while (Number > 0)
	{
		Remainder = Number % 10;
		Number = Number / 10;
		cout << Remainder << endl;
	}
}
```

```
// Solution #6/2
// Sum of Digits      1234 => 10

int SumOfDigits(int Number)
{
	int Sum = 0, Remainder = 0;

	while (Number > 0)
	{
		Remainder = Number % 10;
		Number = Number / 10;
		Sum = Sum + Remainder;
	}

	return Sum;
}
```

```
// Solution #7/2
//يعكس العدد بالكامل          1234 => 4321

int ReverseNumber(int Number)
{
	int Remainder = 0, Number2 = 0;

	while (Number > 0)
	{
		Remainder = Number % 10;
		Number = Number / 10;
		Number2 = Number2 * 10 + Remainder;
	}

	return Number2;
}
```

```cpp
// Solution #8/2
// Digit Frequency

// Input:
// 1223222
// 2

// Output:
// Digit 2 Frequency is 5 Time(s).

int CountDigitFrequency(short DigitToCheck, int Number)

{
	int FreqCount = 0, Remainder = 0;
	while (Number > 0)
	{
		Remainder = Number % 10;
		Number = Number / 10;

		if (DigitToCheck == Remainder)
		{
			FreqCount++;
		}
	}
	return FreqCount;
}
```

```cpp
// Solution #9/2
// Digit Frequency

// Input:
// 1223222

// Output:
// Digit 1 Frequency is 1 Time(s).
// Digit 2 Frequency is 5 Time(s).
// Digit 3 Frequency is 1 Time(s).

void PrintAllDigitsFrequencey(int Number)
{
	cout << endl;

	for (int i = 0; i < 10; i++)
	{
		short DigitFrequency = 0;

		DigitFrequency = CountDigitFrequency(i, Number);
		if (DigitFrequency > 0)
		{
			cout << "Digit " << i << " Frequencey is "
				<< DigitFrequency << " Time(s).\n";
		}
	}
}
```

```
// Solution #10/2
// Print Digits In Order

// Input: 1234

// Output:
// 1
// 2
// 3
// 4

void PrintDigits(int Number)
{
	int Remainder = 0;

	while (Number > 0)
	{
		Remainder = Number % 10;
		Number = Number / 10;
		cout << Remainder << endl;
	}
}
```

```
Solution #11/2
Palindrome Number

NOTE: Palindrome is a number that reads the same from right to left.

// Input:
// 1234
// 12321

// Output:
// No, it is NOT a Palindrome number.
// Yes, it is a Palindrome number.

bool IsPalindromeNumber(int Number)
{
	return Number == ReverseNumber(Number);
}
```

```
// Solution #12/2
// Inverted Number Pattern

// Input:                  Input:
// 3                       5

// Output:                 Output:
// 333                     55555
// 22                      4444
// 1                       333
//                         22
//                         1

void PrintInvertedNumberPattern(int Number)
{
	cout << "\n";

	for (int i = Number; i >= 1; i--)
	{
		for (int j = 1; j <= i; j++)
		{
			cout << i;
		}

	cout << "\n";
	}
}
```

```
// Solution #13/2
// Number Pattern

// Input:                  Input:
// 3                       5

// Output:                 Output:
// 1                       1
// 22                      22
// 333                     333
// 4444
// 55555

void PrintNumberPattern(int Number)
{
	cout << "\n";
	for (int i = 1; i <= Number; i++)
	{
		for (int j = 1; j <= i; j++)
		{
			cout << i;
		}
		cout << "\n";
	}
}
```

```cpp
// Solution #14/2
// Inverted Letter Pattern

// Input:                Input:
// 3                     5

// Output:               Output:
// CCC                   EEEEE
// BB                    DDDD
// A                     CCC
//                       BB
//                       A

void PrintInvertedLetterPattern(int Number)
{
      cout << "\n";
      for (int i = 65 + Number - 1; i >= 65; i--)
      {
            for (int j = 1; j <= Number - (65 + Number - 1); j++)
            {
                  cout << char(i);
            }

      cout << "\n";
      }
}
```

```cpp
// Solution #15/2
// Letter Pattern

// Input:                Input:
// 3                     5

// Output:               Output:
// A                     A
// BB                    BB
// CCC                   CCC
// DDDD
// EEEEE

void PrintLetterPattern(int Number)
{
      cout << "\n";

      for (int i = 65; i <= 65 + Number - 1; i++)
      {
            for (int j = 1; j <= i - 65 + 1; j++)
            {
                  cout << char(i);
            }

      cout << "\n";
      }
}
```

```cpp
// Solution #16/2
// All Words from AAA to ZZZ


// Output:
// AAA
// AAB
// AAC
// AAD
// .
// .
// .
// ZZZ

void PrintWordsFromAAAtoZZZ()
{
      cout << "\n";
      string word = "";

      for (int i = 65; i <= 90; i++)
      {
            for (int j = 65; j <= 90; j++)
            {
                  for (int k = 65; k <= 90; k++)
                  {
                        word = word + char(i);
                        word = word + char(j);
                        word = word + char(k);

                        cout << word << endl;

                        word = "";
                  }
            }

            cout << "\n____________________________\n";
      }
}
```

```
// Solution #17/2
// Guess a 3-Letter Password

// Input:
// AAF

// Output:
// Trial[1]: AAA
// Trial[2]: AAB
// Trial[3]: AAC
// Trial[4]: AAD

// Password is AAD
// Found after 6 Trial(s)
string ReadPassword()
{
	string Password;
	cout << "Please enter a 3-Letter Password (all capital)?\n";
	cin >> Password;
	return Password;
}

bool GuessPassword(string OriginalPassword)
{
	string word = "";
	int Counter = 0;

	cout << "\n";
	for (int i = 65; i <= 90; i++)
	{
		for (int j = 65; j <= 90; j++)
		{
			for (int k = 65; k <= 90; k++)
			{
				Counter++;

				word = word + char(i);
				word = word + char(j);
				word = word + char(k);

				cout << "Trial [" << Counter << "] : " << word
<< endl;

				if (word == OriginalPassword)
				{

					cout << "\nPassword is " << word << "\n";
					cout << "Found after " << Counter << "
Trial(s)\n";

					return true;
				}
				word = "";
			}
		}
	}
	return false;
}
```

```cpp
// Solution #18/2
// Encrypt/Decrypt Text

// Text Before Encryption: Mohammed
// Text After Encryption: Oqjcoogf
// Text After Decryption: Mohammed

string ReadText()
{
	string Text;
	cout << "Please enter Text?\n";
	getline(cin, Text);
	return Text;
}

string EncryptText(string Text, short EncryptionKey)
{
	for (int i = 0; i <= Text.length(); i++)
	{
		Text[i] = char((int)Text[i] + EncryptionKey);
	}
	return Text;
}

string DecryptText(string Text, short EncryptionKey)
{
	for (int i = 0; i <= Text.length(); i++)
	{
		Text[i] = char((int)Text[i] - EncryptionKey);
	}
	return Text;
}

int main() {
	const short EncryptionKey = 2; //this is the key.

	string TextAfterEncryption, TextAfterDecryption;
	string Text = ReadText();
	TextAfterEncryption = EncryptText(Text, EncryptionKey);
	TextAfterDecryption = DecryptText(TextAfterEncryption,
EncryptionKey);

	cout << "\nText Before Encryption : ";
	cout << Text << endl;
	cout << "Text After Encryption : ";
	cout << TextAfterEncryption << endl;
	cout << "Text After Decryption : ";
	cout << TextAfterDecryption << endl;

	return 0;
}
```

```cpp
// Solution #19/2
// Random 3 numbers from 1 to 10

// 9
// 5
// 3

#include <iostream>
#include<cstdlib>
using namespace std;

int RandomNumber(int From, int To)
{
	//Function to generate a random number
	int randNum = rand() % (To - From + 1) + From;
	return randNum;
}

int main()
{
	//Seeds the random number generator in C++, called only once
	srand((unsigned)time(NULL));

	// cout << rand() << endl;
	// cout << rand() << endl;

	cout << RandomNumber(1, 10) << endl;
	cout << RandomNumber(1, 10) << endl;
	cout << RandomNumber(1, 10) << endl;

	return 0;
}
```

```
// Solution #20/2
// Random Small Letter, Capital Letter, Special Character, and Digit
in order

// Output:

// i
// G
// $
// 7

enum enCharType { SamallLetter = 1, CapitalLetter = 2,
SpecialCharacter = 3, Digit = 4 };

char GetRandomCharacter(enCharType CharType)
{
      switch (CharType)
      {
            case enCharType::SamallLetter:
            {
                  return char(RandomNumber(97, 122));
                  break;
            }
            case enCharType::CapitalLetter:
            {
                  return char(RandomNumber(65, 90));
                  break;
            }
            case enCharType::SpecialCharacter:
            {
                  return char(RandomNumber(33, 47));
                  break;
            }
            case enCharType::Digit:
            {
                  return char(RandomNumber(48, 57));
                  break;
            }
      }
}

int main()
{
      //Seeds the random number generator in C++, called only once
      srand((unsigned)time(NULL));

      cout << GetRandomCharacter(enCharType::SamallLetter) << endl;
      cout << GetRandomCharacter(enCharType::CapitalLetter) << endl;
      cout << GetRandomCharacter(enCharType::SpecialCharacter) <<
endl;
      cout << GetRandomCharacter(enCharType::Digit) << endl;

      return 0;
}
```

```cpp
// Solution #21/2
// Generate Keys

// Output:
// Key [1]: TQST-MAKQ-ZJSD-QSKW
// Key [2]: XVIF-RBDD-TASQ-RKRZ
// Key [3]: TOOY-HXMH-HSWZ-ELQO
// Key [4]: IXZE-UIYB-HKAZ-AUOU


// Output: TQST
string GenerateWord(enCharType CharType, short Length)
{
	string Word;

	for (int i = 1; i <= Length; i++)
	{

		Word = Word + GetRandomCharacter(CharType);

	}
	return Word;
}


// Output: TQST-MAKQ-ZJSD-QSKW
string GenerateKey()
{
	string Key = "";

	Key = GenerateWord(enCharType::CapitalLetter, 4) + "-";
	Key = Key + GenerateWord(enCharType::CapitalLetter, 4) + "-";
	Key = Key + GenerateWord(enCharType::CapitalLetter, 4) + "-";
	Key = Key + GenerateWord(enCharType::CapitalLetter, 4);

	return Key;
}

// Key [1]: TQST-MAKQ-ZJSD-QSKW
// Key [2]: XVIF-RBDD-TASQ-RKRZ
// Key [3]: TOOY-HXMH-HSWZ-ELQO
void GenerateKeys(short NumberOfKeys)
{
	for (int i = 1; i <= NumberOfKeys; i++)
	{
		cout << "Key [" << i << "] : ";
		cout << GenerateKey() << endl;
	}
}
```

```
// Solution #33/2
// Fill Array With Keys

// Input:
// 5

// Output:
// Key [0]: TQST-MAKQ-ZJSD-QSKW
// Key [1]: XVIF-RBDD-TASQ-RKRZ
// Key [2]: TOOY-HXMH-HSWZ-ELQO
// Key [3]: IXZE-UIYB-HKAZ-AUOU
// Key [4]: MATF-PRQA-PQEQ-TRIM

void FillArrayWithKeys(string arr[100], int arrLength)
{
	for (int i = 0; i < arrLength; i++)
		arr[i] = GenerateKey();
}


void PrintStringArray(string arr[100], int arrLength)
{
	cout << "\nArray elements:\n\n";
	for (int i = 0; i < arrLength; i++)
	{
		cout << "Array[" << i << "] : ";
		cout << arr[i] << "\n";
	}
	cout << "\n";
}
```

```cpp
// Solution #22/2
// Repeated Elements Count In Array

// Input:
// 5

// Enter array elements:
// Element[1]: 1
// Element[2]: 1
// Element[3]: 1
// Element[4]: 2
// Element[5]: 3

// Enter the number you want to check:
// 1

// Output:
// Original array: 1 1 1 2 3

// Number 1 is repeated 3 time(s)

void ReadArray(int arr[100], int& arrLength)
{
      cout << "\nEnter number of elements:\n";
      cin >> arrLength;

      cout << "\nEnter array elements: \n";
      for (int i = 0; i < arrLength; i++)
      {
            cout << "Element [" << i + 1 << "] : ";
            cin >> arr[i];
      }
      cout << endl;
}

void PrintArray(int arr[100], int arrLength)
{
      for (int i = 0; i < arrLength; i++)
            cout << arr[i] << " ";

      cout << "\n";
}

int TimesRepeated(int Number, int arr[100], int arrLength)
{
      int count = 0;
      for (int i = 0; i <= arrLength - 1; i++)
      {
            if (Number == arr[i])
            {
                  count++;
            }
      }
      return count;
}
```

```cpp
// Solution #23/2
// Fill Array With Random Numbers from 1 to 100

// Input:
// 10

// Output:
// Array Elements: 56 55 83 71 32 52 17 28 71 52

void FillArrayWithRandomNumbers(int arr[100], int& arrLength)

{
      cout << "\nEnter number of elements:\n";
      cin >> arrLength;

      for (int i = 0; i < arrLength; i++)
            arr[i] = RandomNumber(1, 100);
}

void FillArrayWithRandomNumbers(int arr[100], int arrLength)
{
      for (int i = 0; i < arrLength; i++)
            arr[i] = RandomNumber(1, 100);
}
```

```cpp
// Solution #24/2
// Max of Random Array

// Input:
// 10

// Output:
// Array Elements: 65 91 54 42 75 32 53 57 57 30

// Max Elements: 91

int MaxNumberInArray(int arr[100], int arrLength)
{
      int Max = 0;

      for (int i = 0; i < arrLength; i++)
      {
            if (arr[i] > Max)
            {
                  Max = arr[i];
            }
      }
      return Max;
}
```

```cpp
// Solution #25/2
// Min of Random Array

// Input:
// 10

// Output:
// Array Elements: 30 99 72 47 95 67 29 13 80 64

// Min Elements: 13

int MinNumberInArray(int arr[100], int arrLength)
{
	int Min = 0;
	Min = arr[0];

	for (int i = 0; i < arrLength; i++)
	{
		if (arr[i] < Min)
		{
			Min = arr[i];
		}
	}
	return Min;
}
```

```cpp
// Solution #26/2
// Sum of Random Array

// Input:
// 10

// Output:
// Array Elements: 10 6 22 40 12 75 23 32 25 75

// Sum of all number is: 320

int SumArray(int arr[100], int arrLength)

{
	int Sum = 0;

	for (int i = 0; i < arrLength; i++)
	{
		Sum += arr[i];
	}

	return Sum;
}
```

```
// Solution #27/2
// Average of Random Array

// Input:
// 10

// Output:
// Array Elements: 26 36 44 73 8 56 98 67 33 71

// Average of all number is: 51


float ArrayAverage(int arr[100], int arrLength)
{
      return (float)SumArray(arr, arrLength) / arrLength;
}
```

```
// Solution #28/2
// Copy Array

// Input:
// 10

// Output:
// Array 1 elements:
// 1 47 51 18 85 62 51 61 82 4

// Array 2 elements after copy:
// 1 47 51 18 85 62 51 61 82 4


void CopyArray(int arrSource[100], int arrDestination[100], int
arrLength)
{
      for (int i = 0; i < arrLength; i++)
            arrDestination[i] = arrSource[i];
}
```

```
// Solution #29/2
// Copy only prime numbers

// Input:
// 10

// Output:
// Array 1 elements:
// 42 68 35 1 70 25 79 59 63 65

// Prime Numbers in Array2:
// 1 79

void CopyOnlyPrimeNumbers(int arrSource[100], int arrDestination[100],
int arrLength, int& arr2Lenght)
{
	int Counter = 0;
	for (int i = 0; i < arrLength; i++)
	{
		if (CheckPrime(arrSource[i]) == enPrimNotPrime::Prime)
		{
			arrDestination[Counter] = arrSource[i];
			Counter++;
		}
	}

	arr2Lenght = --Counter;
}
```

```
// Solution #30/2
// Sum of 2 Arrays to a third one

// Input:
// How many elements?
// 10

// Output:
// Array 1 elements:
// 42 68 35 1 70 25 79 59 63 65

// Array 2 elements:
// 6 46 82 28 62 92 96 43 28 37

// Sum of array1 and array2 elements:
// 48 114 117 29 132 117 175 102 91 102

void SumOf2Arrays(int arr1[100], int arr2[100], int arrSum[100], int
arrLength)
{
	for (int i = 0; i < arrLength; i++)
	{
		arrSum[i] = arr1[i] + arr2[i];
	}
}
```

```
// Solution #31/2
// Shuffle Ordered Array

// Input:
// 10

// Output:
// Array elements before shuffle:
// 1 2 3 4 5 6 7 8 9 10

// Array elements after shuffle:
// 2 4 6 3 10 1 7 8 5 9

void SwapNumbers(int& A, int& B)
{
	int Temp;

	Temp = A;
	A = B;
	B = Temp;
}

void FillArrayWith1toN(int arr[100], int arrLength)
{
	for (int i = 0; i < arrLength; i++)
		arr[i] = i + 1;
}

void ShuffleArray(int arr[100], int arrLength)
{
	for (int i = 0; i < arrLength; i++)
	{
		SwapNumbers(arr[RandomNumber(1, arrLength) - 1],
arr[RandomNumber(1, arrLength) - 1]);
	}
}
```

```cpp
// Solution #32/2
// Copy Array In Reverse Order

// Input:
// 10

// Output:
// Array 1 elements:
// 64 8 62 19 2 21 15 74 96 85

// Array 2 elements after copying array 1 in reversed order:
// 85 96 74 15 21 2 19 62 8 64

void CopyArrayInReverseOrder(int arrSource[100], int
arrDestination[100], int arrLength)
{
      for (int i = 0; i < arrLength; i++)
            arrDestination[i] = arrSource[arrLength - 1 - i];
}
```

```cpp
// Solution #34/2
// Return number index in array

// Input:
// Enter number of elements:
// 10

// Output:
// Array 1 elements:
// 90 89 62 62 91 47 60 67 60 67

// Please enter a number to search for?
// 62

// Number you are looking for is: 62
// The number found at position: 2
// The number found its order: 3

short FindNumberPositionInArray(int Number, int arr[100], int
arrLength)
{
      /*This function will search for a number in array and return its
index,
      or return -1 if it does not exists*/

      for (int i = 0; i < arrLength; i++)
      {
            if (arr[i] == Number)
                  return i;// and return the index
      }

      //if you reached here, this means the number is not found
      return -1;
}
```

```cpp
void PrintNumberPosition(int Number, int arr[100], int arrLength)
{
	short NumberPosition = FindNumberPositionInArray(Number, arr,
arrLength);

	if (NumberPosition == -1)
		cout << "The number is not found :-(\n";
	else
	{
		cout << "The number found at position: ";
		cout << NumberPosition << endl;
		cout << "The number found its order: ";
		cout << NumberPosition + 1 << endl;
	}
}

int main()
{
	//Seeds the random number generator in C++, called only once
	srand((unsigned)time(NULL));

	int arr[100], arrLength = 0;

	FillArrayWithRandomNumbers(arr, arrLength);

	cout << "\nArray 1 elements:\n";
	PrintArray(arr, arrLength);

	int Number = ReadNumber();
	cout << "\nNumber you are looking for is: " << Number << "\n";

	PrintNumberPosition(Number, arr, arrLength);

	return 0;
}
```

```
// Solution #35/2
// Check Number In Array

// Input:                                     Input:
// 10                                         10

// Output:                                    Output:
// Array 1 elements:                          Array 1 elements:
// 83 92 66 21 62 96 40 13 19 30              83 92 66 21 62 96 40 13 19 30

// Please enter an number to search for?      Please enter an number to search for?
// 66                                         66

// Number you are looking for is: 66          Number you are looking for is: 66
// Yes, The number is found :-)               No, The number is not found :-(


bool IsNumberInArray(int Number, int arr[100], int arrLength)

{
	return FindNumberPositionInArray(Number, arr, arrLength) != -1;
}

void PrintFoundOrNotFound(int Number, int arr[100], int arrLength)
{
	// main يمكنك كتابة فقط هذه الجملة من الكود أو الكود كاملاً في فانكشون الرئيسي

	cout << "\nNumber you are looking for is: " << Number << endl;

	if (!IsNumberInArray(Number, arr, arrLength))
		cout << "No, The number is not found :-(\n";
	else
	{
		cout << "Yes, it is found :-)\n";
	}
}

int main()
{
	//Seeds the random number generator in C++, called only once
	srand((unsigned)time(NULL));

	int arr[100], arrLength = 0;

	FillArrayWithRandomNumbers(arr, arrLength);

	cout << "\nArray 1 elements:\n";
	PrintArray(arr, arrLength);

	int Number = ReadNumber();

	PrintNumberFoundOrNotFoundInArray(Number, arr, arrLength);

	return 0;
}
```

```
// Solution #36/2
// Add Array Element Simi Dynamic


// Please enter a number? 10

// Do you want to add more numbers? [0]:No, [1]:yes? 1

// Please enter a number? 20

// Do you want to add more numbers? [0]:No, [1]:yes? 1

// Please enter a number? 30

// Do you want to add more numbers? [0]:No, [1]:yes? 0

// Array Length: 3
// Array elements: 10 20 30

void AddArrayElement(int Number, int arr[100], int& arrLength)
{
      //its a new element so we need to add the length by 1
      arrLength++;
      arr[arrLength - 1] = Number;
}

void InputUserNumbersInArray(int arr[100], int& arrLength)
{
      bool AddMore = true;

      do
      {
            AddArrayElement(ReadNumber(), arr, arrLength);
            cout << "\nDo you want to add more numbers? [0]:No,[1]:yes?
";
            cin >> AddMore;

      } while (AddMore);
}
```

```
// Solution #37/2
// Resolve Problem 28/2 Copy Array Using (AddArrayElement)

void CopyArrayUsingAddArrayElement(int arrSource[100], int
arrDestination[100], int arrLength, int& arrDestinationLength)
{
      for (int i = 0; i < arrLength; i++)
            AddArrayElement(arrSource[i], arrDestination,
arrDestinationLength);
}
```

```
// Solution #38/2
// Copy Odd Numbers to a new Array

// Input:
// 10

// Output:
// Array 1 elements:
// 59 14 84 36 6 51 48 91 96 67

// Array 2 Odd numbers:
// 59 51 91 67

void CopyOddNumbers(int arrSource[100], int arrDestination[100], int
arrLength, int& arrDestinationLength)
{
      for (int i = 0; i < arrLength; i++)
            if (arrSource[i] % 2 != 0)
            {
                  AddArrayElement(arrSource[i], arrDestination,
arrDestinationLength);
            }
}
```

```
// Solution #39/2
// Copy Prime Numbers to a new Array

// Input:
// 10

// Output:
// Array 1 elements:
// 61 100 32 75 81 95 50 98 13 70

// Array 2 Prime numbers:
// 61 13

void CopyPrimeNumbers(int arrSource[100], int arrDestination[100], int
arrLength, int& arrDestinationLength)
{
      for (int i = 0; i < arrLength; i++)
            if (CheckPrime(arrSource[i]) == enPrimNotPrime::Prime)
            {
                  AddArrayElement(arrSource[i], arrDestination,
arrDestinationLength);
            }
}
```

```
// Solution #40/2
// Copy Distinct Numbers to Array

// Output:
// Array 1 elements:
// 10 10 10 50 50 70 70 70 70 90

// Array 2 distinct elements:
// 10 50 70 90

NOTE: Distinct

void FillArray(int arr[100], int& arrLength)
{
      // Hard coded array elements
      arrLength = 10;
      arr[0] = 10;
      arr[1] = 10;
      arr[2] = 10;
      arr[3] = 50;
      arr[4] = 50;
      arr[5] = 70;
      arr[6] = 70;
      arr[7] = 70;
      arr[8] = 70;
      arr[9] = 90;
}

void CopyDistinctNumbersToArray(int arrSource[100], int
arrDestination[100], int SourceLength, int& DestinationLength)
{
      for (int i = 0; i < SourceLength; i++)
      {
            if (!IsNumberInArray(arrSource[i], arrDestination,
DestinationLength))
            {
                  AddArrayElement(arrSource[i], arrDestination,
DestinationLength);
            }
      };
}
```

```cpp
// Solution #41/2
// Is Palindrome Array

// Input:
// 10 20 30 30 20 10

// Output:
// Array Elements:
// 10 20 30 30 20 10

// Yes array is Palindrome

void FillArray(int arr[100], int& arrLength)
{
      // 10 20 30 30 20 10

      arrLength = 6;
      arr[0] = 10;
      arr[1] = 20;
      arr[2] = 30;
      arr[3] = 30;
      arr[4] = 20;
      arr[5] = 10;
}

bool IsPalindromeArray(int arr[100], int Length)
{
      for (int i = 0; i < Length; i++)
      {
            if (arr[i] != arr[Length - i - 1])
            {
                  return false;
            }
      };
      return true;
}


int main()
{
      int arr[100], Length = 0;

      if (IsPalindromeArray(arr, Length))
            cout << "\nYes array is Palindrome\n";
      else
            cout << "\nNO array is NOT Palindrome\n";
      return 0;
}
```

```
// Solution #42/2
// Count Odd Numbers In Array

// Input:
// 10

// Output:
// Array Elements: 60 78 15 49 56 6 4 3 21 23

// Odd Numbers count is: 5

int OddCount(int arr[100], int arrLength)
{
	int Counter = 0;
	for (int i = 0; i < arrLength; i++)
	{
		if (arr[i] % 2 != 0)
		{
			Counter++;
		}
	}
	return Counter;
}
```

```
// Solution #43/2
// Count Even Numbers In Array

// Input:
// 10

// Output:
// Array Elements: 14 92 70 15 58 76 84 62 10 43

// Even Numbers count is: 8

int EvenCount(int arr[100], int arrLength)
{
	int Counter = 0;
	for (int i = 0; i < arrLength; i++)
	{
		if (arr[i] % 2 == 0)
		{
			Counter++;
		}
	}
	return Counter;
}
```

```
// Solution #44/2
// Count Positive Numbers In Array

// Input:
// 10

// Output:
// Array Elements: 17 -9 -9 -90 -72 -100 -4 60 -84 -15

// Positive Numbers count is: 2

int PositiveCount(int arr[100], int arrLength)
{
      int Counter = 0;
      for (int i = 0; i < arrLength; i++)
      {
            if (arr[i] >= 0)
            {
                  Counter++;
            }
      }
      return Counter;
}
```

```
// Solution #45/2
// Count Negative Numbers In Array

// Input:
// 10

// Output:
// Array Elements: -58 -16 -21 -4 -91 82 98 66 -8 -33

// Negative Numbers count is: 7

int NegativeCount(int arr[100], int arrLength)
{
      int Counter = 0;
      for (int i = 0; i < arrLength; i++)
      {
            if (arr[i] < 0)
            {
                  Counter++;
            }
      }
      return Counter;
}
```

```cpp
// Solution #46/2
// MyABS

// Input:

// -10

// Output:
// My abs Result: 10
// C++ abs Result: 10
float MyABS(float Number)
{
    if (Number > 0)
        return Number;
    else
        return Number * -1;
}
```

```cpp
// Solution #47/2
// MyRound

// Input:                      Input:                  Input:
// 10.7                        10.3                    -10.7

// Output:                     Output:                 Output:
// My Round Result: 11         My Round Result: 10     My Round Result: -11
// C++ round Result: 11        C++ round Result: 10    C++ round Result: -11


float GetFractionPart(float Number)
{
    return Number - int(Number);
}

int MyRound(float Number)
{
    int IntPart;
    IntPart = int(Number);

    float FractionsPart = GetFractionPart(Number);

    if (FractionsPart >= .5)
    {
        if (Number > 0)
        {
            return ++IntPart;
        }
        else
        {
            return --IntPart;
        }
    }
    else
    {
        return IntPart;
    }
}
```

```
// Solution #48/2
// MyFloor
// Input:                      Input:                  Input:
// 10.7                        10.3                    -10.3

// Output:                     Output:                 Output:
// My MyFloor Result: 10       My MyFloor Result: 10   My MyFloor Result: -11
// C++ floor Result: 10        C++ floor Result: 10    C++ floor Result: -11


int MyFloor(float Number)
{
      if (Number > 0)
            return int(Number);
      else
            return int(Number) - 1;
}
```

```
// Solution #49/2
// MyCeil

// Input:                      Input:
// 10.7                        -10.7

// Output:                     Output:
// My MyCeil Result: 11        My MyCeil Result: -10
// C++ ceil Result: 11         C++ ceil Result: -10

int MyCeil(float Number)
{
      if (abs(GetFractionPart(Number)) > 0)
      {
            if (Number > 0)
                  return int(Number) + 1;
            else
                  return int(Number);
      }
      else
      {
            return int(Number);
      }
}
```

```cpp
// Solution #50/2
// MySqrt

// Input:
// 10.7

// Output:
// My MySqrt Result: 11
// C++ sqrt Result: 11


int MySqrt(float Number)
{
	return pow(Number, 0.5);
}
```