**American International University Bangladesh (AIUB)**

****

**Faculty of science & Technology**

**Department of Computer Science**

**LAB MANUAL-02**

CSC 2211: Algorithms

|  |
| --- |
| **TITLE** |

**Algorithmic and Computational Thinking**

|  |
| --- |
| **PREREQUISITE** |

* To be able to write, build and run a C++ program in CodeBlocks.
* To be able to identify and understand the basic components of a C++ program
* Able to write user define function and to use the library function (e.g. math.h, rand()).
* Good knowledge to use Array/List as argument of functions.
* Good knowledge to use pointers as argument of functions and function return a pointer.
* Function call by reference and call by value (swap example)
* Dynamic memory allocation in C++
* Standard Template library(STL in C++)

|  |
| --- |
| **OBJECTIVE** |

* Algorithmic and Computational Thinking

|  |
| --- |
| **Step by step learn C++** |

* Download Code Blocks that is open source, cross platform, free C, C++ and Fortran IDE.
* C++ data types and variables
* Operator and expression
* Conditional structure (if, if else, if else if…. Else, switch)
* Loops(for, while, do while loop)
* Functions
* Array
* Pointer
* Function, array and pointer
* String
* Structure and class
* Object Oriented programming

**Function, Array and Pointer in C++**

#include<iostream>

using namespace std;

int main()

{

double a, b, c;

cout<<"Enter two number: ";

cin>>a>>b;

c = a+b;

cout<<"the result: "<<c;

return 0;

}

|  |
| --- |
| **LAB WORK** |

Write the following algorithm in C++

1. The greatest common divisor (GCD) of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers. For example, the GCD of 8 and 12 is 4, that is, gcd(8,12)=4. Euclid's algorithm is given in the pseudo code. Write the following algorithm in recursively.
   * + 1. ***input n,m***
       2. ***if n = m then STOP***
       3. ***if n is greater than m: do n = n-m***
       4. ***else do m = m-n***
       5. ***GOTO 2***
       6. ***print n or m***
2. Consider the following algorithm:
   * + 1. ***input n***
       2. ***print n***
       3. ***if n = 1 then STOP***
       4. ***if n is odd then n ←− 3n + 1***
       5. ***else n ←− n/2***
       6. ***GOTO 2***

Given the input 22, the following sequence of numbers will be printed

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that 0 < n < 1, 000, 000 (and, in fact, for many more numbers than this.) Given an input n, it is possible to determine the number of numbers printed before and including the 1 is printed. For a given n this is called the cycle-length of n. In the example above, the cycle length of 22 is 16

1. Find the maximum product of two distinct numbers in a sequence of non-negative integers.  
   **Input:** A sequence of non-negative integers.  
   **Output:** The maximum value that can be obtained by multiplying two different elements from sequence.

**Naive Approach**

Given a sequence of non-negative integers compute

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5 | 6 | 2 | 7 | 4 |
| 5 |  | 30 | 10 | 35 | 20 |
| 6 | 30 |  | 12 | 42 | 24 |
| 2 | 10 | 12 |  | 14 | 8 |
| 7 | 25 | 42 | 14 |  | 28 |
| 4 | 20 | 24 | 8 | 28 |  |

***MaxPairwiseProductNaive(A[1:::n]):***

***product=0***

***for i from 1 to n:***

***for j from 1 to n:***

***if i != j:***

***if product < A[i] · A[j]:***

***product A[i] · A[j]***

***return product***

1. Fast Approach

***MaxPairwiseProductFast(A[1:::n]):  
 index1=1  
 for i from 2 to n:  
 if A[i] > A[index1]:  
 index1=i***

***index2=1  
 for i from 2 to n:  
 if A[i]!=A[index1] and A[i] > A[index2]:  
 index2=i  
return A[index1] · A[index2]***