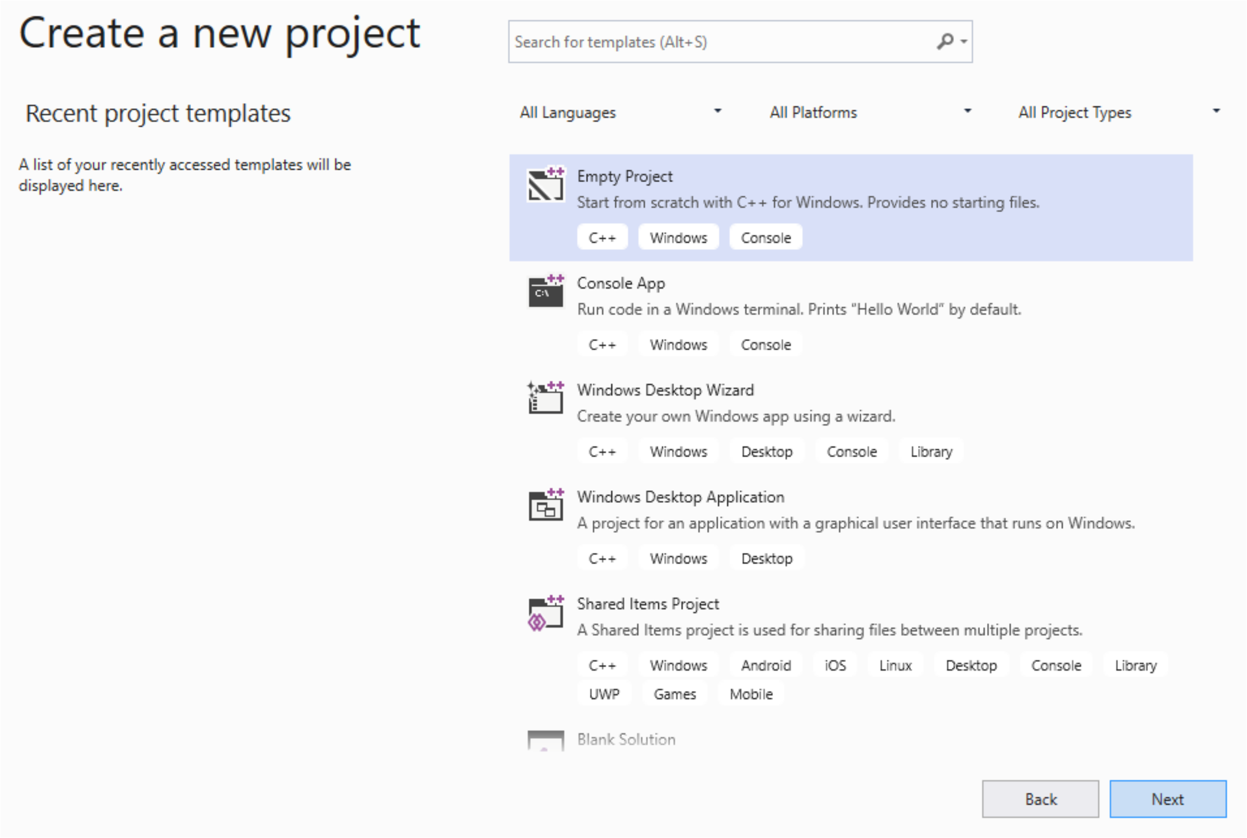
PRACTICE GUIDELINES

(WEEK 1)

➢ <u>Running a single file in project</u>

- *Create an empty project*



Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Empty Project
Start from scratch with C++ for Windows. Provides no starting files.

Console App
Run code in a Windows terminal. Prints "Hello World" by default.

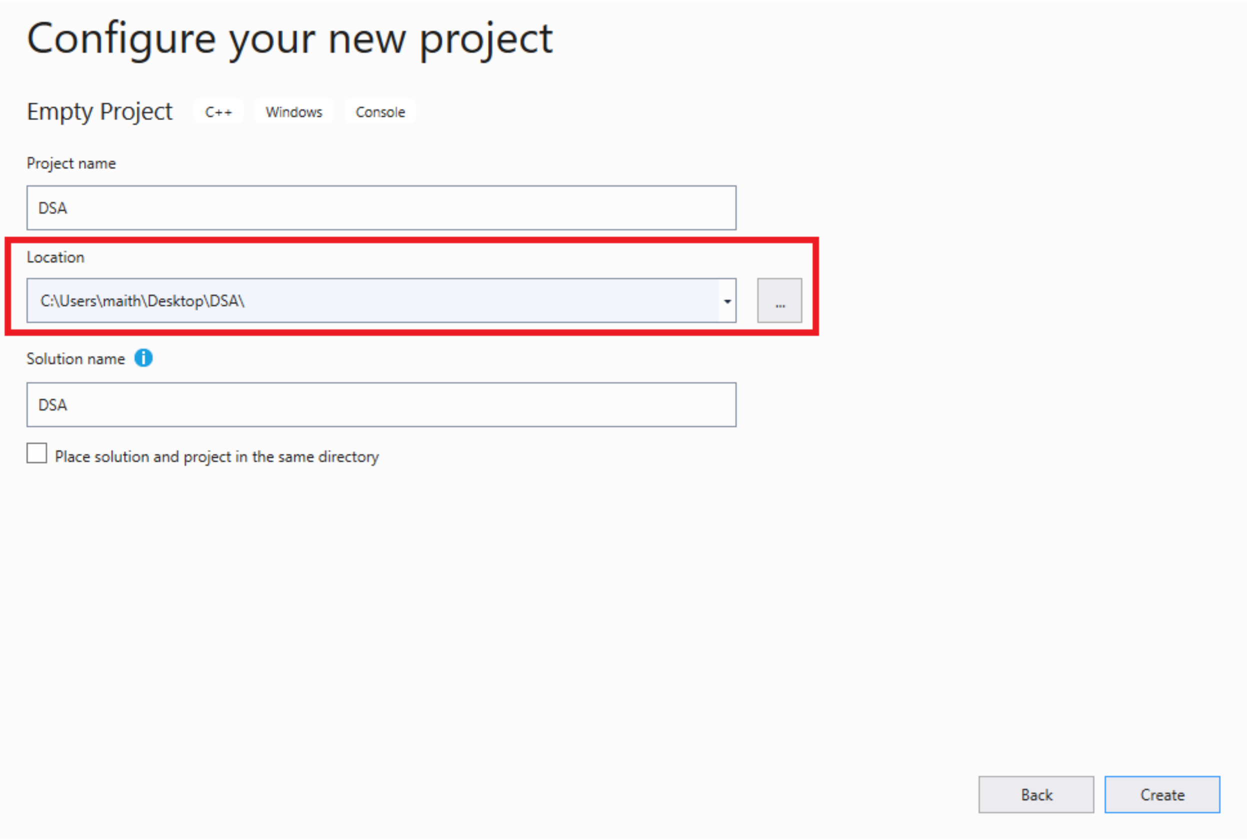
Windows Desktop Wizard
Create your own Windows app using a wizard.

Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.

Shared Items Project
A Shared Items project is used for sharing files between multiple projects.

Blank Solution

- *Set your project location*



Configure your new project

Empty Project

Project name: DSA

Location: C:\Users\maith\Desktop\DSA\

Solution name: DSA

□ Place solution and project in the same directory

- *Copy your existing cpp files into the project root.*
- *Add existing item to Source Files of project.*
- *Define your cpp file with content:*

```
#include "DSA/DSA/Header.h"
#include <stdio.h>
#include <conio.h>
#include <string.h>

#ifdef __Demkytucuachuoi__
int main()
{
}
#endif
```

- *Create header file with content:*

```
#include <iostream>
#define __Demkytucuachuoi__
```

Exercise 6

- *Store the remainder when the number is divided by 2 in an array.*
- *Divide the number by 2*
- *Repeat the above two steps until the number is greater than zero.*
- *Print the array in reverse order now.*

For Example:

If the binary number is 10.

Step 1*: Remainder when 10 is divided by 2 is zero. Therefore, arr[0] = 0.*

Step 2*: Divide 10 by 2. New number is 10/2 = 5.*

Step 3*: Remainder when 5 is divided by 2 is 1. Therefore, arr[1] = 1.*

Step 4*: Divide 5 by 2. New number is 5/2 = 2.*

Step 5*: Remainder when 2 is divided by 2 is zero. Therefore, arr[2] = 0.*

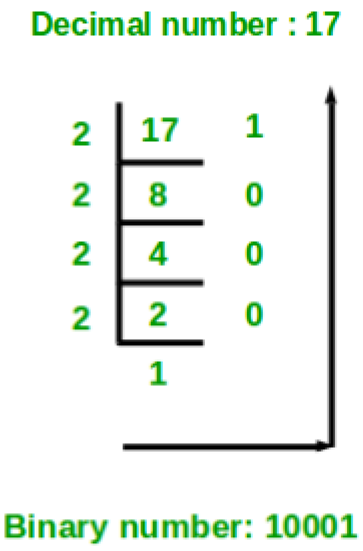
Step 6*: Divide 2 by 2. New number is 2/2 = 1.*

Step 7*: Remainder when 1 is divided by 2 is 1. Therefore, arr[3] = 1.*

Step 8*: Divide 1 by 2. New number is 1/2 = 0.*

Step 9*: Since number becomes = 0. Print the array in reverse order. Therefore the equivalent binary number is 1010.*

Below diagram shows an example of converting the decimal number 17 to equivalent binary number.



- **<u>Exercise 11</u>**

To define a number of elements in the array, using:

int n = sizeof(arr) / sizeof(arr[0]);

sizeof is a C/C++ function that returns the number of bytes of storage used by the parameter. In this case, sizeof(arr) return the number of bytes held by the entire structure, sizeof(arr[0]) however returns the number of bytes a single element of the array uses for storage. And given that in C/C++, elements of an array must be of the same type, and thus have the same number of bytes, doing a simple division results in the number of items stored in that array.

- **<u>Exercise 12</u>**

floor(x): *Returns the largest integer that is smaller than or equal to x (i.e : rounds downs the nearest integer).*

Input : 2.5 Output : 2

Input : -2.1 Output : -3

Input : 2.9 Output : 2

ceil(x): *Returns the smallest integer that is greater than or equal to x (i.e : rounds up the nearest integer).*

Input : 2.5 Output : 3

Input : -2.1 Output : -2

Input : 2.9 Output : 3

A naive approach *is to take an element and compare with all other elements and if it is greater then increment the count and then check if count is greater than n/2 elements then print.*

An efficient method *is to sort the array in ascending order and then print last ceil(n/2) elements from sorted array.*

- **Exercise 13**

Method 1 (Simple)

The naive approach is to run two loops and check one by one element of array check that array elements have at-least two elements greater than itself or not. If its true then print array element.

Time Complexity: O(n^2*)*

Method 2 (Use Sorting)

We sort the array first in increasing order, then we print first n-2 elements where n is size of array.

Time Complexity: O(n Log n)

Method 3 (Efficient)

In second method we simply calculate second maximum element of array and print all element which is less than or equal to second maximum.

Time Complexity: O(n)

- ## Exercise 14

Naive approach: *A simple approach is to store the odd numbers first, one by one till N and then storing the even numbers one by one till N, and then printing the kth element.*

Efficient approach: *Find the index where the first even element will be stored in the generated array. Now if the value of k is less then or equal to index then the desired number will be k * 2 – 1 else the desired number will be (k – index) * 2*

- ## Exercise 15

Find a pair with maximum product in array of Integers Given an array with both +ive and -ive integers, return a pair with highest product.

Examples: Input: arr[] = {1, 4, 3, 6, 7, 0} Output: {6,7}
Input: arr[] = {-1, -3, -4, 2, 0, -5} Output: {-4,-5}

A Simple Solution *is to consider every pair and keep track maximum product.*

Time Complexity : O(n^2)

A Better Solution *is to use sorting. Below are detailed steps.*

1) Sort input array in increasing order.

2) If all elements are positive, then return product of last two numbers.

3) Else return maximum of products of first two and last two numbers.

Time complexity of this solution is O(nLog n).

An Efficient Solution *can solve the above problem in single traversal of input array. The idea is to traverse the input array and keep track of following four values.*

a) Maximum positive value

b) Second maximum positive value

c) Maximum negative value i.e., a negative value with maximum absolute value

d) Second maximum negative value.

At the end of the loop, compare the products of first two and last two and print the maximum of two products.

Time complexity: O(n)

Auxiliary Space: O(1)