Name: Deepanshi
Student ID#: 456807 Email ID: d.807@mybvc.ca

# SODV 1101
# Programming Fundamentals

Winter 2024
Assignment 2

**Instructor: Mahbub Murshed** Date: 2024-04-03

**Directions:**

- Written questions can be submitted via soft copy or a scanned document and submitted by d2l dropbox, **as a single doc file or single pdf file**. Answers should be written clearly and coherently. **Programming questions must be submitted in repl.it.**
- Assignments must be submitted by the posted due date, or be subject to the course's late policy. Work must be completed individually and in accordance with the academic integrity policy.
- All codes shown in this assignment are compiled and tested except for questions that aim to find errors in a C++ program.
- For all written questions, you may assume that the usual header files are included as follows:

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;
```

**Note: The assignment's written questions not submitted as a single file will not be graded!**

| **Written Questions** | |
|---|---|
| **1** | / 6 |
| **2** | / 14 |
| **3** | / 10 |
| **Programming Questions** | |
| **Quality** | / 20 |
| **Functionality** | / 50 |
| **Total** | |
| | / 100 |

# Assignment 2: Written Questions

1. Give concise answers to the following questions

a) What is an array? / 3

ANS: An **array** is a data structure that is used to store multiple values of similar data types in a contiguous memory location.

**For example**, if we have to store the marks of 4 or 5 students then we can easily store them by creating 5 different variables but what if we want to store marks of 100 students or say 500 students then it becomes very challenging to create that numbers of variable and manage them. Now, arrays come into the picture that can do it easily by just creating an array of the required size.

**syntax:** data_type array_name[Size_of_array];
**Example:** int a[5];
Indexing of an array starts from **0.** It means the first element is stored at the 0th index, the second at 1st, and so on.
a[5]=

| | | | | |
|---|---|---|---|---|
| 2 | 4 | 6 | 8 | 10 |
| a[0] | a[1] | a[2] | a[3] | a[4] |

b) What do the break and continue keywords do? / 3

**ANS:** **BREAK:** Break statements in loops are used to terminate the loop.
**Example:** int main(){
```
    int i;
    cout << "The loop with break produces output as:
\n";
    for (i = 1; i <= 5; i++) {
// Program comes out of loop when
// i becomes multiple of 3.
    if ((i % 3) == 0)
      break;
    else
   cout << i << " ";
}
```
**Output:** The loop with break produces output as:
```
1 2
```

**CONTINUE:** Continue statements are somewhat similar to break statements but the main difference between the continue and break is that break terminate the loop entirely and **continue terminates only the current iteration.**
**Example:** int main(){
```
    int i;
    cout << "The loop with break produces output as:
\n";
    for (i = 1; i <= 5; i++) {
// Program comes out of loop when
// i becomes multiple of 3.
    if ((i % 3) == 0)
      break;
    else
   cout << i << " ";
}
```

**OUTPUT:** The loop with continue produces output as:

1 2 4 5

2.

a) What is the exact output of the code below if we execute the function call `q2(20)`? / 4

```
void q2(int n)
{
    for (int i = 1; i < n; i *= 2)
        cout << i;
}
```

**code:**

ANS:

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

void q2(int n) {
  for (int i = 1; i < n; i *= 2)
     cout << i;
}
int main() {
  q2(20);
  return 0;
}
```

**Output:** 124816

The output is 1,2,4,8,16 but on console it looks like 124816 because we do not print the space between the numbers ( cout << i;)

**Steps:**

1. `i` starts at 1.
2. In the first iteration, **`i` is less than 20, so it prints 1.** Then `i` is doubled to 2.
3. In the second iteration, **`i` (now 2) is still less than 20, so it prints 2**. Then `i` is doubled to 4.
4. In the third iteration, **`i` (now 4) is still less than 20, so it prints 4**. Then `i` is doubled to 8.

5. In the fourth iteration, **`i` (now 8) is still less than 20, so it prints 8**. Then `i` is doubled to 16.
6. In the fifth iteration, **`i` (now 16) is still less than 20, so it prints 16**. Then `i` is doubled to 32.
7. Since **32 is greater than or equal to 20**, the loop terminates.

b) The function in part (a) uses a for-loop. Rewrite the function using a while-loop without changing how the code works. / 5

ANS: **CODE using a while loop:**
function in while loop is written as:

```
void q2(int n) {
  int i = 1;
  while (i < n) {
      cout << i;
      i *= 2;
  }
 }
```

**Full code:**

```
#include <iostream>
using namespace std;
void q2(int n) {
  int i = 1;
  while (i < n) {
    cout << i;
    i *= 2;
  }
}
int main() {
  q2(20);
  return 0;
}
```

**OUTPUT:** 124816

The output is 1,2,4,8,16 but on console it looks like 124816 because we do not print the space between the numbers ( cout << i;)

**Steps:**

1. We **start** the program **by calling the `main()` function.**
2. Inside `main()`, we **call the `q2(20)`** function.

3. In `q2(20)`, we **start with `i = 1`** and print `1` since it's less than `20`.
4. Then, we **double `i` to get 2** `(i*=2)`, and print it too.
5. We keep **doubling `i` until it's greater than or equal to `20`**, printing each value.
6. Finally, the **loop stops** when **`i` becomes `32`**, that is greater than `20`.
7. We **return to `main()`** after `q2(20)` finishes.
8. The **program ends** with a successful exit status (`0`).
9. On Console, we get **`124816`**, that are the values of `i` printed during each step of the function.

c) Rewrite the function from part (a) using a do-while loop. (Hint: make sure the code produces the same output for all cases). / 5

**ANS: CODE using a do-while loop:**
function in do-while loop is written as:

```
void q2(int n) {
   int i = 1;
   do {
       cout << i;
       i *= 2;
   } while (i < n);
}
```

**Full Code:**

```
#include <iostream>
using namespace std;
void q2(int n) {
  int i = 1;
  do{
    cout << i;
    i *= 2;
  }while (i < n);
}
int main() {
  q2(20);
  return 0;
}
```

**OUTPUT:** 124816

The output is 1,2,4,8,16 but on console it looks like 124816 because we do

not print the space between the numbers ( cout << i;)
Steps:

1. `do { ... } while (i < n);` This is a `do-while` loop. It means that the loop body will be executed at least once, and then the condition `i < n` will be checked. If the condition is true, the loop will continue; otherwise, it will terminate.
2. Inside the loop:
   - `cout << i;` Prints the current value of `i`.
   - `i *= 2;` Doubles the value of `i`.

3. In the following function implementation, there are 2 syntax errors and 2 logic errors. Mark the errors in the code below with a circled number corresponding to the spaces provided. Briefly explain why it is an error **and** give a fix for it. / 10

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

int isAscending(int[] arr, int size) { /* declaration of []arr in c++ is wrong it should be arr[]*/
            i = 1;   //Declared and initialized variable 'i'.
            for (; i <= size; i++)    //Changed loop condition to 'i < size'.
            {
                        if (arr[i - 1] > arr[i]) {
                                    return false;
                        }
                        else
                                    return true; /*Moved the return statement outside the loop to ensure
all elements are checked.*/
            }
}
```

1) **Syntax Error:** i= 1; Declared and initialized variable 'i' i.e int i =1;

2) **Syntax Error:** []arr; the declaration of array is wrong; in C++, it is declared as arr[]

3) **Logic Error:** for (; i <= size; i++); The loop condition should be 'i < size' instead of 'i <= size' to prevent accessing out-of-bound memory.

When **i<= size** it will pass the value 0 if the array is in ascending order, so the **i< size** it will pass the value 1 if the array is in ascending order.

4) **Logic Error:** declaration of **return true;** it should be declared outside the for loop; if the first comparison (arr[i - 1] > arr[i]) is not true, the function immediately returns true without checking the rest of the array. This is incorrect because it implies that the array is in ascending order as long as the first pair of elements satisfies the condition.

**Corrected function:**

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

int isAscending(int arr[], int size) {
   int i = 1;
   for (; i < size; i++) {
      if (arr[i - 1] > arr[i]) {
         return false;
      }
   }
   return true;
}
```

**also, what if we use boolean data type instead of integer**

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

bool isAscending(int arr[], int size) { /*Added 'bool' to specify the return type and
corrected the syntax for the array parameter.*/
   int i = 1;
   for (; i < size; i++) {
      if (arr[i - 1] > arr[i]) {
         return false;
      }
   }
   return true;
}
```

If I use `int` instead of `bool` as the return type for the `isAscending` function. When `int is used` as the return type, the function will return either `0` or `1` instead of `false` or `true`. In C++, `0` typically represents `false` and any non-zero value represents `true`.

In C++, `bool` is represented as `true` and `false`, but when printed to the console, they are internally converted to integers `1` and `0`, respectively. So, whether we use `int` or `bool` as the return type, we will still see `0` and `1` printed on the console.

# Program 2(A): Array Merging
## (Please submit in repl.it)

**Learning Objective**

- Identify program structure and control the flow of execution.
- Write structured programming instructions so that the computer shows input/output behaviors
- Use different types of loops to write code with the required number of iterations.
- Demonstrate logical thinking by using programming syntax and strategies.
- Use arrays to store collections of data.

## Overview

Write a program to merge two arrays

## Directions

For example, consider two arrays A and B.
A = {10, 25, 35, 40, 55}
B = {15, 30, 5, 20, 45, 65}

The merged array should be another array C with elements {5, 10, 15, 20, 25, 30, 35, 40, 45, 55, 65}. The resultant merged array should contain all the elements of A and B in sorted order. This doesn't mean that you can two copy two arrays to the third array and sort the resultant array. Initially, you can sort Array 'A' and Array 'B' with your own sorting method. Afterward, you will need to merge the contents of the two arrays into a single array by taking individual values from the two arrays one by one and inserting them into the resultant Array 'C' in the appropriate position, so that it becomes sorted right after the insert operation.

**Sample Input/output:**

How many elements are there in the first Array? 5
How many elements are there in the second Array? 6
Elements in ' A': 10, 25, 35, 40, 55 (Randomly populated)
Elements in ' B': 15, 30, 5, 20, 45, 65 (Randomly populated)
Sorted Elements in ' A': 10, 25, 35, 40, 55 (Sorted by calling your own created method)
Sorted Elements in ' B': 5, 15, 20, 30, 45, 65 (Sorted by calling your own created method)
Elements in 'C': 5, 10, 15, 20, 25, 30, 35, 40, 45, 55, 65 (After Merging A and B; and no sorting methods called)

**Solution:**

// Firstly, we need

// How many elements are there in the first Array? // How many elements are there in the second Array?

```
#include <cstdlib>

#include <iostream>

using namespace std;

int main() {

int num1, num2;

cout << "How many elements are there in the first Array?" << endl;

cin >> num1;

cout << "How many elements are there in the second Array?" << endl;

cin >> num2;

return 0;

}
```

**//** Then elements should be generated randomly and display **(function)**

```
void randomPopulate(int x[], int y)

{

for (int i = 0; i < y; i++)

{

x[i] = rand() % 100 + 1;

}

}

void display(int x[], int y)

{

for (int i = 0; i < y; i++)

{

cout << x[i] << ", ";
```

```
}

}

int main()

{

int num1, num2;

srand(7);/*run program with srand(7) before generating random numbers, we will get the
same sequence of random numbers.*/

int A[num1];
randomPopulate(A, num1);
cout << "Elements in 'A': ";
display(A, num1);
int B[num2];

srand(8);
randomPopulate(B, num2);
cout << endl << "Elements in 'B': ";
display(B, num2);
return 0;
}

//sorting

//one loop for position

// one loop for checking and sorting the numbers

//function for sorting

void arraySort(int x[], int y)

{

int temp;

for (int i = 0; i < y-1; i++)

{

for (int j = i + 1; j < y; j++)

{

        if (x[j] < x[i])

                {
                        temp = x[i];

                        x[i] = x[j];

                        x[j] = temp;

                }

}

}

}
```

//Now, merge the array

//merging function

```
void arrayMerge(int x[], int y[], int z[], int n1, int n2, int n3) { int i = 0; int j = 0; int k = 0;

// Merging sorted arrays A and B into C
while (i < n1 && j < n2)//This loop iterates until either array1 or array2 is fully traversed.
{
   if (x[i] < y[j]){
      z[k++] = x[i++];
   }
   else{
      z[k++] = y[j++];
   }
}
 // Copy remaining elements of first array.
while (i < n1){
   z[k++] = x[i++];
}
 // Copy remaining elements of element array.
while (j < n2){
   z[k++] = y[j++];
}
}
```

# Program 2(B): Replace String
## (Please submit in repl.it)

**Learning Objective**

- Identify program structure and control the flow of execution.
- Write structured programming instruction so that the computer shows input/output behaviors
- Use for loops to write code with a required number of iterations.
- Build programs that can construct or parse strings.
- Create formatted outputs by using strings
- Demonstrate logical thinking by using programming syntax and strategies.

## Overview

Write a C++ program that takes a single line input string. Your program also collects a search string and a replace string. Finally, your program should search for all occurrences of the given search string in the single-line input and replaces them all with the replace string.

## Directions

In the single-line input, there can be two or more occurrences of the same search string. The search string can be followed by a space character (' '), comma (','), or full stop character ('.'). Therefore, use a loop, inside the loop at every iteration replace one search string with the replace string. Repeat this process until all the search strings are replaced.

**Sample Input/output 01:**

```
Single Line Input String:
Alex is a good student. Programming is fun for Alex.
Search String:
Alex
Replace String:
Your Name
Your Name is a good student. Programming is fun for Your Name.
```

**Solution:**

```
  int sstrLength = sstr.length();
  int lineLength = line.length();
```

/***Here, it calculates the lengths of the search string (`sstr`) and the input string (`line`) using the `length()` function of the string class and stores them // in variables `sstrLength` and `lineLength` respectively.***/

```
  // Use a for loop to iterate through the input string
  for (int i = 0; i <= lineLength - sstrLength; ++i) {
```

/*T**his initiates a for loop to iterate through the characters of the input string. The loop starts from `i = 0` and continues until `i` reaches a position where it's impossible for the search string to fit in the remaining portion of the input string. This is done to avoid unnecessary comparisons.***/

```
    // Check if the substring starting at position i matches the search string
    if (line.substr(i, sstrLength) == sstr) {
```

/* **Within the loop, it checks if the substring of the input string starting at position `i` and of length `sstrLength` matches the search string (`sstr`). It does so by using the `substr()` function, which extracts a substring from the input string.***/

```
      // If it matches, check if it's a complete word (not part of another word)
      if ((i == 0 || !isalpha(line[i - 1])) && (i + sstrLength == lineLength || !isalpha(line[i +
sstrLength]))) {
```

/***If the substring matches the search string, it checks whether it constitutes a complete word. This is done by examining the characters before and after the matched substring. The condition `(i == 0 || !isalpha(line[i - 1]))` checks if the character before the matched substring is either the beginning of the input string or not an alphabet character (which would indicate the start of a word). Similarly, the condition `(i + sstrLength == lineLength || !isalpha(line[i + sstrLength]))` checks if the character after the matched substring is either the end of the input string or not an alphabet character.** */

```
        // If it's a complete word, replace the substring with the replace string
        line.replace(i, sstrLength, rstr);
        // Move i to the next position after the replaced string
        i += rstr.length() - 1;
        // Adjust the length of the input string after replacement
        lineLength = line.length();
```

/***If the matched substring is indeed a complete word, it replaces it with the replace string (`rstr`) using the `replace()` function. Then it moves the index `i` to the next position after the replaced string and updates the length of the input string (`lineLength`)*/**