A. Y. 2022 – 2023

FIRST AND SECOND SEMESTER

PRACTICE FINAL EXAM

CIS 1101 – Programming 1

(Part 2)

Name:
Program and Year:
CIS 1101 Group #:
Date:

NOTES:

I. You have **150 minutes** to take the test. Kindly inspect the exam time duration carefully and do not spend too much time on a single question.

II. Each test has different directions. Follow them carefully.

III. Answer each item by typing your answers in the fields and spaces provided.

IV. Always try to write accurately using appropriate and efficient program logic and proper coding syntax. For programming problems, follow the coding conventional rules that are set for the class (e.g. all conditional statements must involve relational operators; break and continue statements shall be only used in switch blocks.).

V. Usage of compilers, calculators, and/or related applications is **STRICTLY NOT ALLOWED**.

VI. Should you have a concern on any of the questions, kindly contact or message the examiner for inspection or clarification.

VII. Answer each question as far as you can. Try not to leave blanks.

VIII. Once you are finished with the exam, save the PDF file with your answers and submit it as an attachment to the examiner's email address (20100215@usc.edu.ph) or through direct message for checking.

PROBLEM #1: Compound Interest (10 pts)

Complete the program by writing the code of function **computeCompoundInterest().**

This function takes the following as parameters: The principal amount, the interest rate (e.g. 5% -> 0.05), and an integer representing the number of years. It will compute for and return the total interest generated given the indicated amount compounded annually at the given rate for the given time period.

ASSUMPTIONS/CONSTRAINTS:

Number of years, interest rates, and the number of years will be all positive numbers.

Do not alter the existing code.

EXAMPLE 1:

```
Input principal: 10000
Input rate (percent): 5
Input number of years: 5

Total interest in 5 years: P 2762.82
```

EXAMPLE 2:

```
Input principal: 100
Input rate (percent): 8
Input number of years: 2

Total interest in 2 years: P 16.62
```

EXAMPLE 2:

```
Input principal: 10000
Input rate (percent): 11
Input number of years: 3

Total interest in 3 years: P 367.63
```

```
#include<stdio.h>

//complete the function prototype
___ computeCompoundInterest(____);

void main(){
   float p;
   int rate, years;
   ___ answer;

   printf("Input principal: ");
   scanf("%f",&p);
   printf("Input rate (percent): ");
   scanf("%d",&rate);
   printf("Input number of years: ");
   scanf("%d",&years);

   //complete the function call
   answer =

   //print the final line of the display here


}

___ computeCompoundInterest(____){
   //write your code here
}
```

PROBLEM #2: Updating Load Balance (15 pts)

Complete the program by writing the code of **function updateBalance().**

This function takes a person's existing load balance and the number of persons he needs to contact as parameters. For every person, he does a minute of calls which will cost him 7.50 pesos. He also has the option to subscribe to an Unli Call promo which will cost him 25 pesos to have as many calls as he want should the total cost exceed 25 pesos. Calls cannot be made with an insufficient balance.

The function will update the given balance by deducting the amount spent and then return the actual number of persons that are able to be called.

If the number of persons called does not match the requested number of people to call, add another display "Insufficient balance".

CONSTRAINTS:

Do not alter the existing code.

EXAMPLE 1:

```
Input balance: 30
Input number of persons to call: 5

Updated balance: P 5.00
Persons called: 5
```

EXAMPLE 2:

```
Input balance: 20.75
Input number of persons to call: 5

Updated balance: P 5.75
Persons called: 2
Insufficient balance to call everyone
```

EXAMPLE 3:

```
Input balance: 100
Input number of persons to call: 3

Updated balance: P 77.50
Persons called: 3
```

```
#include<stdio.h>

//complete the function prototype
___ updateBalance(____);

void main(){
   float bal;
   int num;
   ___ answer;

   printf("Input balance: ");
   scanf("%f",&p);
   printf("Input number of persons to call: ");
   scanf("%d",&num);

   //complete the function call
   answer =

   //print the final lines of the display here


}

___ updateBalance(____){
   //write your code here
}
```

PROBLEM #3: Give me some bits! (15 pts)

Complete the program by writing the code of **function generateBinaryDigits().**

This function takes a given integer as parameter and converts it to its binary notation. This is done by creating an integer (decimal form) whose digits represent the binary notation of the given integer.

The resulting binary number shall be returned to the calling function.

ASSUMPTIONS/CONSTRAINTS:

The given integer to be converted shall be always positive.

The given number shall be at most 1000.

Do not alter the existing code.

EXAMPLE 1:

```
Input decimal integer: 100

The binary result is 1100100.
```

EXAMPLE 2:

```
Input decimal integer: 1000

The binary result is 1111101000.
```

EXAMPLE 3:

```
Input decimal integer: 0

The binary result is 0.
```

```
#include<stdio.h>

//complete the function prototype
___ generateBinaryDigits(____);

void main(){
   int num;
   ___ answer;

   printf("Input decimal integer: ");
   scanf("%d",&num);

   //complete the function call
   answer =

   //print the final line of the display here


}

___ generateBinaryDigits(____){
   //write your code here
}
```

PROBLEM #4: What comes next? (15 pts)

Complete the program by writing the code of **function getSequenceTerms().**

An arithmetic sequence is one whose succeeding terms are obtained by adding a fixed value to the term before it. Examples are 1,2,3,… and 2,4,6,….

You are given the first three numbers of an arithmetic sequence as well as two more numbers M and N. Your task is to determine and return the Mth to Nth terms of the given sequence stored in a newly created array.

This function takes the following as parameters:

- An array of three integers representing the first three numbers of an arithmetic sequence.
- Two integers M and N to determine the range of the terms to be obtained.
- Pointer to an integer count which stores the number of values in the new array containing the results.

The program will display the returned array containing the obtained terms.

Note that sometimes the three inputs can not form a valid arithmetic sequence, like 1,2,4,…. If that is the case, display a message "Sequence is not valid!" inside the getSequenceTerms() function.

ASSUMPTIONS/CONSTRAINTS:

3 < M < N < 1000

Do not alter the existing code.

EXAMPLE 1:

```
Input first three terms: 2 4 6
Input the nth term ranges: 6 10

Terms 6 to 10 of the sequence are:
{12 14 16 18 20}
```

EXAMPLE 2:

```
Input first three terms: 99 98 97
Input the nth term ranges: 51 55

Terms 51 to 55 of the sequence are:
{59 58 57 56 55}
```

EXAMPLE 3:

```
Input first three terms: 12 15 14
Input the nth term ranges: 11 15

Sequence is not valid!
```

```
#include<stdio.h>
#include<stdlib.h>

//complete the function prototype
___ getSequenceTerms(____);

void main(){
   int terms[3], M, N, count;
   ___ answer;

   printf("Input first three terms: ");
   scanf("%d %d %d", &terms[0], &terms[1], &terms[2]);
   printf("Input the nth term ranges: ");
   scanf("%d %d",&M, &N);


   //complete the function call
   answer =

   //print the final lines of the display here


}

___ getSequenceTerms(____){
   //write your code here
}
```

PROBLEM #5: What's your position? (15 pts)

Complete the program by writing the code of **function getInput() and insertSorted().**

You are given an array whose elements are distinct and should be already sorted. You will now ask for a new number which will be inserted at its proper sorted position in the existing array of numbers.

- If the existing array is not sorted, do not insert the number, then display "Elements are not sorted!"
- If the number already exists in the array, do not insert the number, then display "Element already exists!"
- If insertion is successful, display "Insertion successful!"

The program proceeds to display all the values in the updated array to verify the element has been inserted properly or has not been altered if there are errors.

ASSUMPTIONS/CONSTRAINTS:

There will be always enough room for the new integer to be inserted.

Do not alter the existing code.

EXAMPLE 1:

```
How many numbers: 5
Input numbers: 2 4 6 8 10
Input number to be inserted: 7

Insertion successful!
Array elements: {2 4 6 7 8 10}
```

EXAMPLE 2:

```
How many numbers: 5
Input numbers: 2 4 6 8 0
Input number to be inserted: 7

Elements are not sorted!
Array elements: {2 4 6 7 8 10}
```

EXAMPLE 3:

```
How many numbers: 5
Input numbers: 2 4 6 8 10
Input number to be inserted: 8

Element already exists!
Array elements: {2 4 6 8 10}
```

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100

//complete the function prototypes
___ getInput(____);
___ insertSorted(___);

void main(){
   int arr[MAX], size, num;

   printf("How many numbers: ");
   size = getInput();
   printf("Input numbers: ");
   for(int i=0; i<size; i++){
      arr[i] = getInput();
   }
   printf("Input number to be inserted: ");
   num = getInput();

   //complete the function call for insertSorted()
   ___

   printf("Array elements: {");
   for(int i=0; i<size; i++){
      printf("%d",arr[i]);
      if(i<size-1) printf(" ");
   }
   printf("}");
}
___ getInput(____){
   //write your code here
}

___ insertSorted(____){
   //write your code here
}
```

PROBLEM #6: Are we all part of the family? (15 pts)

Complete the program by writing the code of **function isSubset().**

You are given two arrays A and B.

Each array contains a set of distinct values. You are also given the number of elements from each array.

Your function should determine whether the set B is a subset of the set A.

Set B is a subset of Set A if all the elements in set B are present in Set A.

Your function will return 1 if Set B is a Subset of Set A and 0 if otherwise.

ASSUMPTIONS/CONSTRAINTS:

Sizes of sets A and B must be at least 1.

Size of set B must be <= size of set A.

Do not alter the existing code.

EXAMPLE 1:

```
Input set A size: 5
Input set A elements: 1 2 3 4 5
Input set B size: 3
Input set elements: 3 2 1

Is set B a subset of set A? 1
```

EXAMPLE 2:

```
Input set A size: 5
Input set A elements: 1 2 3 4 5
Input set B size: 3
Input set elements: 3 2 0

Is set B a subset of set A? 0
```

EXAMPLE 3:

```
Input set A size: 5
Input set A elements: 8 6 0 1 2
Input set B size: 5
Input set elements: 0 2 6 8 1

Is set B a subset of set A? 1
```

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100

//complete the function prototypes
void getInputs(int*, int*, char);
___ isSubset(int*, int*, int, int);

void main(){
   int A[MAX], B[MAX], Asize, Bsize;
   ___ answer;

   getInputs(A, &Asize, 'A');
   getInputs(B, &Bsize, 'B');

   //complete the function call
   answer =

   //print the final lines of the display here

}
void getInputs(int arr[], int* size, char ch){
   //complete the code of the function
   printf("Input set %c size: ",ch);
   scanf("%d",____);
   printf("Input set %c elements: ",ch);
   for(int i=0; i<____; i++){
      scanf("%d",&arr[i]);
   }
}

___ isSubset(int A[], int B[], int Asize, int Bsize){
   //write your code here
}
```

PROBLEM #7: How many occurrences? (15 pts)

Complete the program by writing the code of **function createCountArray().**

The function shall accept as parameters an array of integers and its size. It will return a new array that is formatted in such a way that the value of each index X in the array to be returned stores the number of occurrences of the value X found in the given array A.

Example array:

6 5 5 1 4 4 3 3 3 5

The returned array:

Index:	0	1	2	3	4	5	6	7	8	9
Value:	0	1	0	3	2	3	1	0	0	0

CONSTRAINTS:

The values for each element in the given array will range only from 0 to 9.

Do not alter the existing code.

EXAMPLE 1:

```
How many numbers? 10
Input numbers: 6 5 5 5 1 4 4 3 3 3

Count of 0: 0
Count of 1: 1
Count of 2: 0
Count of 3: 3
Count of 4: 2
Count of 5: 3
Count of 6: 1
Count of 7: 0
Count of 8: 0
Count of 9: 0
```

EXAMPLE 2:

```
How many numbers? 10
Input numbers: 1 2 3 1 2 3 1 2 3 9

Count of 0: 0
Count of 1: 3
Count of 2: 3
Count of 3: 3
Count of 4: 0
Count of 5: 0
Count of 6: 0
Count of 7: 0
Count of 8: 0
Count of 9: 1
```

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100

//complete the function prototypes
___ createCountArray(____);

void main(){
   int arr[MAX], size;
   ___ answer;

   printf("How many numbers? ");
   scanf("%d",&size);
   printf("Input numbers: ");
   for(int i=0; i<size; i++){
      scanf("%d",&arr[i]);
   }

   //complete the function call
   answer =

   //complete the code below to display the result as indicated
   printf("\n");
   for(int i=0;i<=9;i++){
      ___
   }
}

___ createCountArray(____){
   //write your code here
}
```