

Final Evaluation: 40%

|  |  |
| --- | --- |
| **Course Identification** | |
|  |  |
| Name of programs – Codes | COMPUTER SCIENCE TECHNOLOGY – PROGRAMMING (420.BP) INFORMATION TECHNOLOGY PROGRAMMER ANALYST – LEA.3Q |
| Course title: | **Algorithms and programming** |
| Course number: | 420-AP1-AS |
| Group: | 07220, 07222 |
| Teacher’s name: | Michelle Khalife |
| Duration: | 3 periods (150 minutes) |
| Semester: | Fall 2020 |
| **Student Identification** | |
| |  |  | | --- | --- | | **Name: Samuel Fauteux** | **Student Number:** **2033158** | | **Date:** December 14th 2020 | **Results:** |   I declare that this is an original work, and that I credited all content sources of which I am not the author (online and printed, images, graphics, films, etc.), in the required quotation and citation style for this work. | |

|  |
| --- |
| **Standard of the Evaluated Competency** |
|  |

**Statement of the evaluated competency – Code**

Use of programming languages – 00Q2

**Evaluated elements of the competency** Analyze the problem  
 Translate the algorithm into a programming language  
 Debug the code  
 Implement the functional test plan

|  |  |  |
| --- | --- | --- |
| ***Competency:*** | Use programming languages - 00Q2 | |
| ***General ministerial and institutional performance criteria:*** | | |
| * Sense of organization * Respect of the code of ethics * Communication skills * Quality of analysis and implementation | | |
| ***Elements of the competency***  ***420.BP (1-2-3 only)*** | | ***Performance criteria specific to each element*** |
| 1. Analyze the problem. | | * 1. Correct breakdown of the problem   2. Proper identification of input and output data and of the nature of the processes   3. Appropriate choice and adaptation of the algorithm |
| 1. Translate the algorithm into a programming language. | | * 1. Appropriate choice of instructions and types of elementary data   2. Efficient modularization of code   3. Logical organization of instructions   4. Compliance with the language syntax   5. Computer code consistent with the algorithm |
| 1. Debug the code. | | * 1. Efficient use of the debugger   2. Identification of all errors   3. Astute choice of debugging strategies   4. Relevance of the corrective actions   5. Clear record of solutions to the problems encountered |
| 1. Implement the functional test plan. | | * 1. Attitudes and behaviours that demonstrate thoroughness   2. Identification of all operational errors   3. Relevance of the corrective actions   4. Proper functioning of the program   5. Clear record of information concerning tests and their results |

|  |
| --- |
| **Instructions** |
|  |
| Permitted equipment: BYOD, VS Code, and/or online editors.Closed book exam, but students may use the dictionary.You may type your code in an editor, but you must copy and paste your answers in-line (following each question) in this booklet. Make sure your code adheres to standards and conventions and is properly indented and highlighted, and not some random unstructured text.No break is allowed during this exam. Students are not allowed to exit the examination room before half of the allotted time has passed. Once a student has exited the classroom, he/she may not re-enter (IPEL – Article 5.12.4).Students must remain silent during the exam.It is the teacher’s responsibility to identify language errors. If such errors are found, teachers may apply a penalty of up to 5% of the grade (IPEL – Article 5.7).Plagiarism, attempts at plagiarism or complicity in plagiarism during a summative evaluation results in a mark of zero (0). In the case of recidivism, in the same course or in another course, the student will be given a grade of '0' for the course in question.(IPEL – Article 5.16). |
| **General guidelines for online reviews:**   * **Log in at least 10 minutes in advance of the exam time.** * **If there is a connection problem, contact the teacher immediately by MIO or TEAMS to inform him/her of the problem. Add a screenshot if possible.** * **Your camera should be open** **at the teacher's request.** * **Headphones are not permitted (except for oral comprehension exercises)** * **Any screenshot sent in place of your work will be refused.** * **Make sure to save the latest version before handing over.** * **Make sure you have sent your exam correctly before the deadline.**   **If we feel that your answers may not be yours, the department reserves the right to complete your evaluation with a virtual meeting to verify that you have reached the required competency.** |
| **Mark Breakdown** |
|  |
| This evaluation is on 100 points, distributed as follows:   |  |  | | --- | --- | | Q1 \_ Algorithm Tracing | For a total of 30 points | | Q2 \_ Functions, Patterns, and 2D arrays | For a total of 30 points | | Q3\_ Parallel Arrays and Structs | For a total of 40 points | |  | **TOTAL: 100 points** |  |  |  | | --- | --- | | Sub-total on 100 (before errors) | /100 | | Penalty for language errors (0.5 point each /maximum 5%) | /100 | | **Evaluation total on 100** | /100 | | **Evaluation total on 40** | /40 | |
|  |
|  |

QUESTION #1 – ALGORITHM TRACING

Consider the following program:

#include <stdio.h>

//

void function1(int array[], int size) {

  for (int step=0; step < size - 1; ++step) {

    //

    int flag = 0;

    //

    for (int i=0; i < size - step - 1; ++i) {

      //

      if (array[i] > array[i+1]) {

        //

        int temp = array[i];

        array[i] = array[i+1];

        array[i+1] = temp;

        flag = 1;

      }

    }

    //

    if (flag == 0)

      break;

  }

}

//   
void function2(int array[], int size) {

  for (int i=0; i<size; ++i)

    printf("%d  ", array[i]);

  printf("\n");

}

int main() {

  int data[] = {2,0,2,0,1,4,9}; // USE YOUR ID HERE, INSTEAD OF THESE NUMBERS

  int size = sizeof(data)/sizeof(data[0]);

  function1(data, size);

  function2(data, size);

  return 0;

}

Use your La Salle College ID to initialize the array. For instance, if your ID is 2020-149, then the array *data* (in main) would look like:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 0 | 2 | 0 | 1 | 4 | 9 |

1. Trace the program from beginning to end. Show your work for at least 3 iterations and draw the array (7-cells table like the one above) every time a change occurs.

I enter the main

The first command becomes: int data[7] = {2,0,3,3,1,5,8}; for me(basically it contains my id)

Size = 7 //the nb of things in the array

I enter function1

Step = 0

Flag = 0

I = 0

If(true)

Temp = 2

Array = 0033158

Array = 0233158

Flag = 1

I = 1

If(false)

I = 2

If(false)

I = 3

If(true)

Temp = 3

Array = 0231158

Array = 0231358

Flag = 1 //again

I = 4

If(false)

I = 5

If(false)

If(false)

Step = 1

Flag = 0

I = 0

If(false)

I = 1

If(false)

I = 2

If(true)

Temp = 3

Array = 0211358

Array = 0213358

Flag = 1

I = 3

If(false)

I = 4

If(false)

If(false)

Step = 2

Flag = 0

I = 0

If(false)

I = 1

If(true)

Temp = 2

Array = 0113358

Array = 0123358

I = 2

If(false)

I = 3

If(false)

If(false)

Step = 3 //if in the inner loop is false all along

If(true) //this is the if in the outer loop

I break from the for and end up at the end of the function so I go back to my main

I enter function2

I = 0

The for prints the array (everything in data, one by one)

Br

I return to my main

Return 0

(so in the end my id was modified, it was printed to the console followed by a break line and the program finished with exit code 0)

1. Look at the series of arrays you’ve drawn and describe what this algorithm did and how the different functions (function1, function2) and variables (e.g., flag, temp, …) work.

Algorithm: sorts my id and then print it to the console followed by a br

Function1: sorts my id

Function2: prints the content of the array followed by a br

Data: is my array in which I originally put my id

Size: the nb of thing in my array

//array and size, the parameter of the two function does the same thing

Array: passes data to the function

Size: passes size to my function //the parameter

Step: tells you in which step of the sorting you are

Flag: is used to check if any change were done to your array during the inner loop

I: is used to refer to the indexes in your array

Temp: is a temporary memory in which you place one of the numbers that will be moved around

1. Fill the 7 comments in the program with appropriate descriptions.

//sorts the array

//will be used to see if you made any change to the array during your inner loop

//compares I to i+1, if I is greater, switches their positions

//the before mentioned comparaison

//temp hold the value of one of the number that are being switched in order for it to not get overwritten

//check if there has been any change in the array during this step

//print the content of the array

1. In terms of *n,* how many iterations does this algorithm need to terminate? 22
2. How would you modify the algorithm so that the final array is printed in reverse?

I would replace the printing for by: for(int I = size – 1; I >= 0; --i)

|  |  |
| --- | --- |
| **Element of competency:** Analyse the problem (00Q2.1) | |
| **Performance criteria** | **Weight** |
| 1.1 Correct breakdown of the problem (#2, #3) | 10 |
| 1.2 Proper identification of input and output data and of the nature of the processes (#1) | 10 |
| 1.3 Appropriate choice and adaptation of the algorithm (#4) | 5 |
| **Element of competency:** Debug the code (00Q2.3) | |
| **Performance criteria** | **Weight** |
| 3.4. Relevance of the corrective actions (#5) | 5 |

QUESTION #2 – FUNCTIONS, PATTERNS & 2D ARRAYS

In the pattern below, each row starts and ends with 1.  
  
The values in between may be computed from the previous row, by adding adjacent pairs of values together. The rule stipulates that, for any non-negative integer *n* and any integer *k*, between 0 and *n,* inclusive: a[n][k] = a[n-1][k-1] + a[n-1][k].

The following pattern is produced when n=6. Notice, as mentioned above, that when *n=6*, you are really going from 0 to 6 (inclusive).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |
| 1 | 2 | 1 |  |  |  |  |
| 1 | 3 | 3 | 1 |  |  |  |
| 1 | 4 | 6 | 4 | 1 |  |  |
| 1 | 5 | 10 | 10 | 5 | 1 |  |
| 1 | 6 | 15 | 20 | 15 | 6 | 1 |

1. Use a 2D array to produce the pattern for n=6

#include <stdio.h>

int main()

{

int n = 6;

int pattern[n][n];

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

{

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

{

if(i = 0)

{

pattern[i][j] = 1;

}

else

{

pattern[i][j] = pattern[i-1][j-1] + pattern[i][j-1];

}

}

}

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

{

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

{

printf("%c", pattern[i][j]);

}

printf("\n");

}

return 0;

}

1. Embed your code in a function that takes *n* as a parameter, and prints this 2D array.

#include <stdio.h>

void DisplayThis(int n)

{

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

{

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

{

printf("%c", pattern[i][j]);

}

printf("\n");

}

}

int main()

{

int n = 6;

int pattern[n][n];

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

{

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

{

if(i = 0)

{

pattern[i][j] = 1;

}

else

{

pattern[i][j] = pattern[i-1][j-1] + pattern[i][j-1];

}

}

}

DisplayThis(n);

return 0;

}

1. Write a function that returns the sum of the numbers in the pattern above. Call this function and display this number.

int DisplayThisNumber(int posX, int posY)

{

if(posX == 0)

{

return 1;

}

return pattern[posX-1][posY-1] + pattern[posX][posY-1];

}

|  |  |
| --- | --- |
| **Element of competency:** Analyse the problem (00Q2.1) | |
| **Performance criteria** | **Weight** |
| 1.1 Correct breakdown of the problem (#1) | 5 |
| 1.2 Proper identification of input and output data and of the nature of the processes (#2, #3) | 10 |
| **Element of competency:** Translate the algorithm into a programming language (00Q2.2) | |
| **Performance criteria** | **Weight** |
| 2.1Appropriate choice of instructions and types of elementary data (#1) | 5 |
| 2.2 Efficient modularization of code (#2) | 5 |
| 2.3 Logical organization of instructions (#3) | 5 |
| 2.4 Compliance with the language syntax | 5 |
| 2.5 Computer code consistent with the algorithm | 5 |

QUESTION #3 – PARALLEL ARRAYS & STRUCTS

The inventory of a shoe store lists shoes by a stock number (range 0-50).

With each stock number there is:

* an associated style name,
* the number of pairs in each size (sizes range from 5 to 11), and
* a price.

1. Give 4 parallel arrays declarations to structure the inventory data.

Int stockNo[50][50];

Char styleName[50][15];

Int nbOfPairPerSize[50][6]; //the second index is the size-5

Float price[50][6]; //the second index is for the size of the shoe

1. Now, give 1 struct declaration to structure the inventory data.

struct Shoe

{

    int stockNo;

    char styleName[15];

    int numberOfPairPerSize[6]; //the i = size - 5

    int price[6];

};

1. How would you enter the following in your struct?
   1. Stock#: 17

shoes[i].stockNo = 17;

* 1. Style name: “Miko”

shoes[i].styleName = {‘M’, ‘i’, ‘k’, ‘o’};

* 1. Sizes 5, 6, 8, 10, and 11, each have 1 item in stock.

shoes[i].numberOfPairPerSize[0] = 1;

shoes[i].numberOfPairPerSize[1] = 1;

shoes[i].numberOfPairPerSize[3] = 1;

shoes[i].numberOfPairPerSize[5] = 1;

shoes[i].numberOfPairPerSize[6] = 1;

* 1. Price: $450.00

shoes[i].price[sizeThaIsThatCosty] = 450;

1. On Black Friday, Miko’s price dropped by 47%. The manager didn’t have the stock number on hand, but he was able to pull the record using the style name (“Miko”) and change the price. Write a functionthat takes two parameters, *style\_name* and *new\_price*, and changes the original price to *new\_price*.
2. void ChangeNameByName(char styleName[], float newPrice)
3. {
4. bool found = 0;
5. for (int i = 0; i < k; ++i)
6. {
7. if(strcmp(shoes[i].styleName, styleName) == 0)
8. {
9. found = 1;
10. shoes[i].price[sizeThatIsThatCosty] = newPrice;
11. }
12. }
14. if(!found)
15. {
16. printf("This style wasn't found in the inventory");
17. }
18. }
19. A client walks in and buys a pair of Miko’s in size 6. How would you reflect this in the inventory?

--shoes[Miko’sId].numberOfPairPerSize[1];

1. How would you display all the inventory?

for (int i = 0; i < k; ++i)

    {

        printf("Stock number: %d\nStyle name: %s\nNumber of pairs\n", shoes[i].stockNo, shoes[i].styleName);

        for (int j = 0; j < 7; ++j)

        {

            printf("Size %d: %d $%.2f\n", j+5, shoes[i].numberOfPairPerSize[j], shoes[i].price[j]);

        }

        printf("\n");

    }

|  |  |
| --- | --- |
| **Element of competency:** Analyse the problem (00Q2.1) | |
| **Performance criteria** | **Weight** |
| 1.1 Correct breakdown of the problem (#1, #2) | 8 |
| 1.2 Proper identification of input and output data and of the nature of the processes (#4) | 4 |
| 1.3 Appropriate choice and adaptation of the algorithm (#5) | 4 |
| **Element of competency:** Translate the algorithm into a programming language (00Q2.2) | |
| **Performance criteria** | **Weight** |
| 2.1Appropriate choice of instructions and types of elementary data (#3) | 4 |
| 2.2 Efficient modularization of code (#6) | 4 |
| 2.3 Logical organization of instructions (#4, #5, #6) | 4 |
| 2.4 Compliance with the language syntax | 1 |
| 2.5 Computer code consistent with the algorithm | 1 |