## STRUCTURED PROGRAMMING

**Username:** ebazan089

## SECTION I: Algorithm/code in C++

### QUESTION 1: DESIGNING THE ALGORITHM /FLOWCHART

*Create the algorithm (pseudo-code) that will ask the user to choose what kind of conversion he/she likes to do (pounds to kilograms or kilograms to pounds).*
*There are 2.2046 pounds in a kilogram, 1000 grams in a kilogram, and 16 ounces in a pound.*
*If the first option was selected, enter a weight in pounds and ounces and outputs the equivalent weight in kilograms and grams or to enter 0 if wants to find out the max, the min and the average of the weights the user enter as far.*
*If the second option was selected, enter a weight in kilograms and grams and outputs the equivalent weight in pounds and ounces or to enter 0 if wants to find out the max, the min and the average of the weights the user enter as far.*
*Include a loop that lets the user repeat this computation for new input values until the user says he or she wants to end the program (the 3th option of the main menu).*

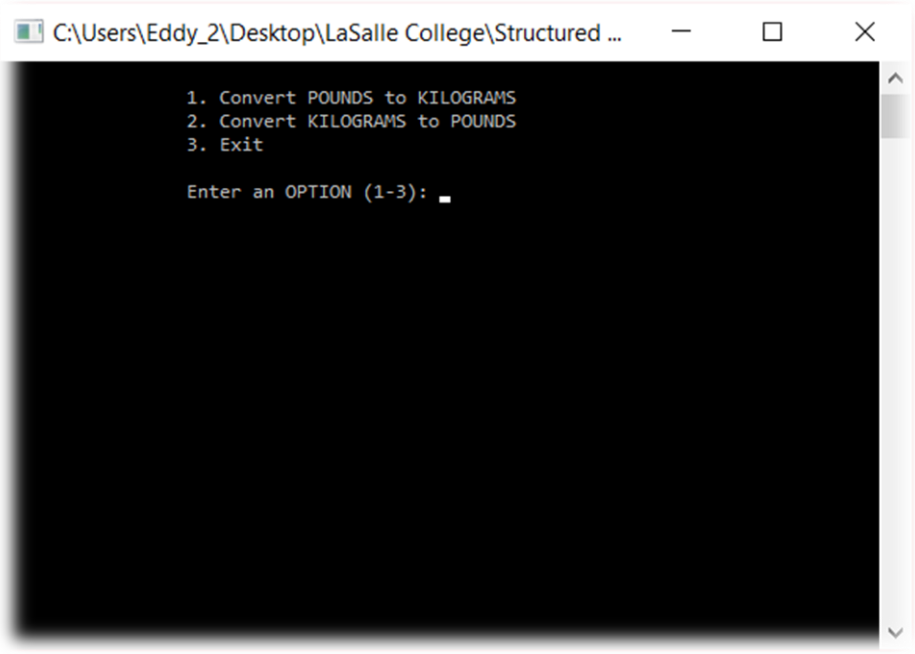
1. **START**
2. Ask the user ENTER one of the three options.
3. If the user choose 1, it will convert pounds to kilograms. ***Go STEP 6***
4. If the user choose 2, it will conver kilograms to pounds. ***Go STEP 7***
5. If the user choose 3, it will leave the program. ***Go STEP 9***
6. User enter the value WEIGHT in POUNDS, then the program will convert this value to KILOGRAMS and it will save in the table. The user can enter until 15 values or it stops when the user enter 0. ***Go STEP 8***
7. User enter the value WEIGHT in KILOGRAMS, then the program will convert this value to POUNDS and it will save in the table. The user can enter until 15 values or it stops when the user enter 0. ***Go STEP 8***
8. The program will show all the values that the user enter, also it will show the MAXIMUM, MINIMUM and AVERAGE. ***Go STEP 2***
9. **END**

**QUESTION 2: TRANSLATING ALGORITHM INTO A C++**

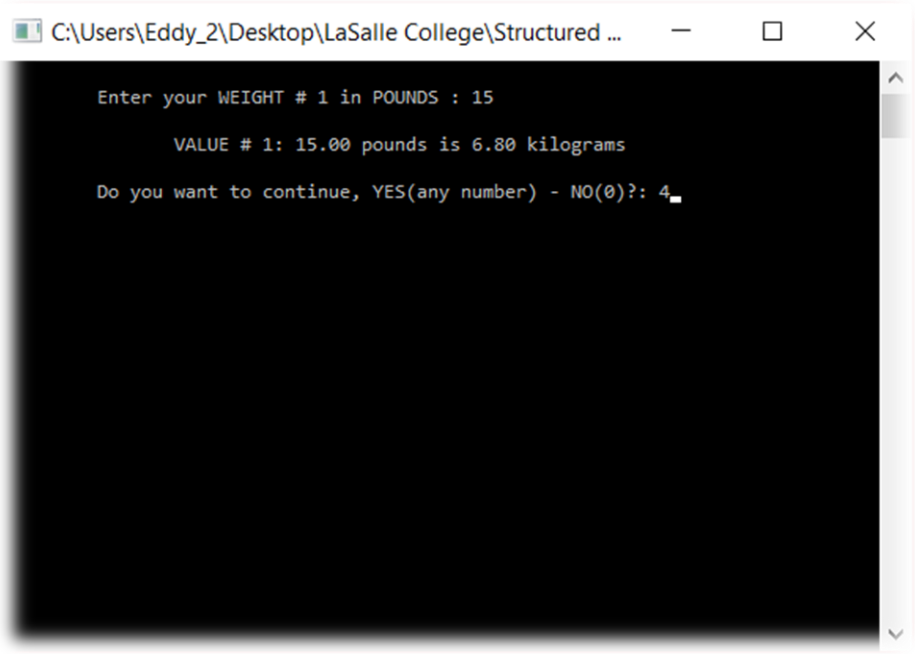
| VARIABLES | DESCRIPTION |
|---|---|
| `int ans, i, x, opt;` | Global variables type INTEGER |
| `double value, maxV=0, minV=0, averageV=0;` | Global variables type DOUBLE |
| `const int SIZE = 15;` | Constant INTEGER with value 15 |
| `double table[SIZE], values[SIZE];` | Array type DOUBLE with the size of the constant SIZE |

| FUNCTION NAME | DESCRIPTION |
|---|---|
| `void pounds_kilograms(double table[]);` | Calculate the value from pounds to kilograms and save it in the table. |
| `void kilograms_pounds(double table[]);` | Calculate the value from kilograms to pounds and save it in the table. |
| `void maxValue(double table[], int tsize, int opt);` | Calculate the MAXIMUM value with all the values of the table. |
| `void minValue(double table[], int tsize, int opt);` | Calculate the MINIMUM value with all the values of the table. |
| `void averageValue(double table[], int tsize, int opt);` | Calculate AVERAGE value with all the values of the table. |

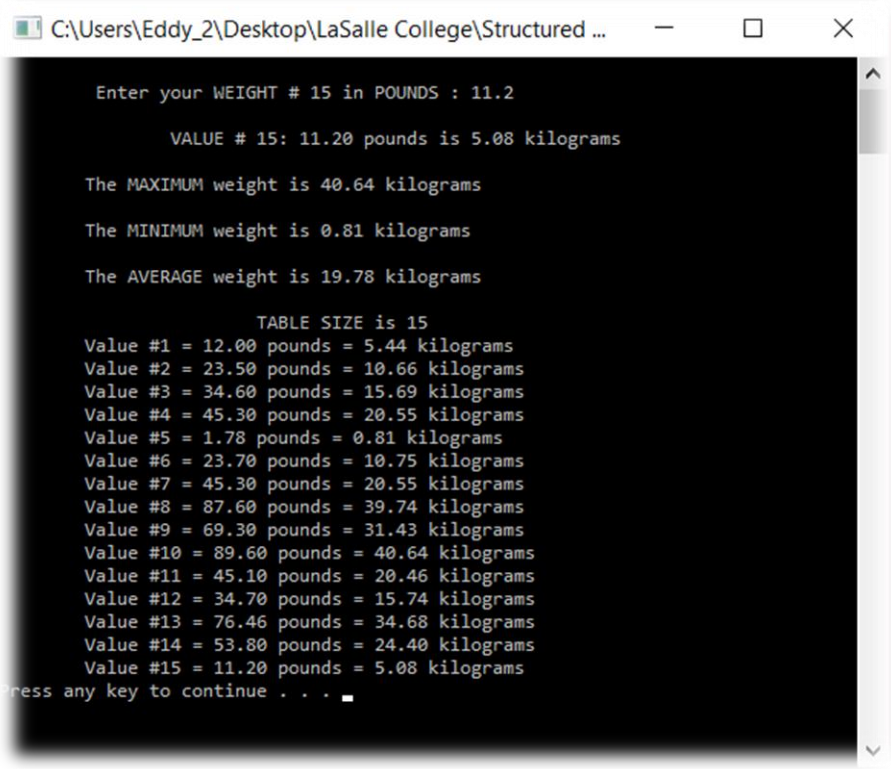
## MENU



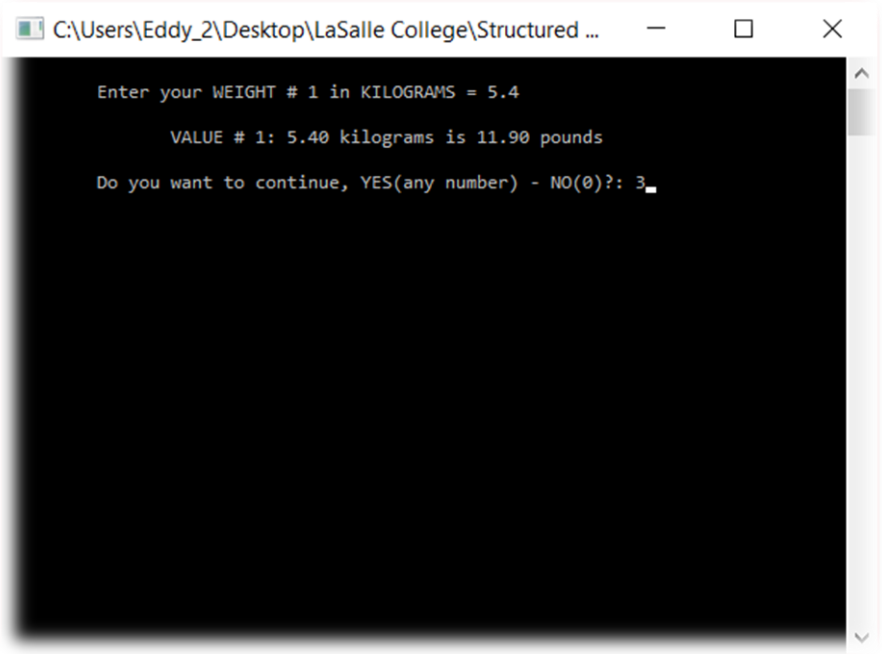

## *OPTION 1:* CONVERT POUNDS TO KILOGRAMS



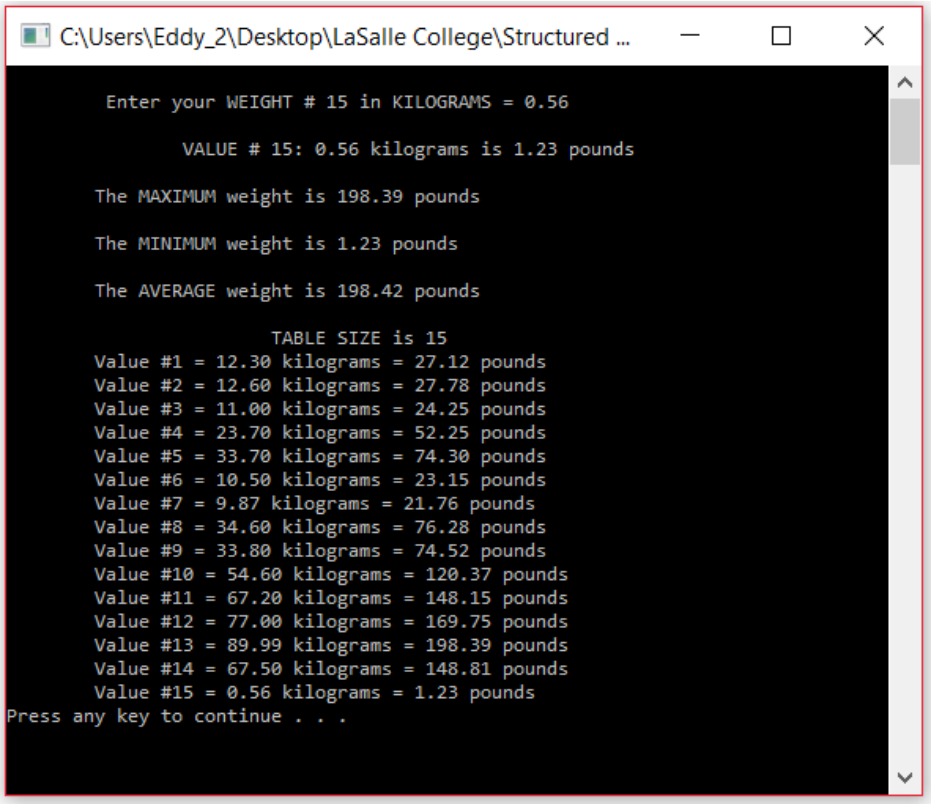
**SHOW *MAXIMUM, MINIMUN AND AVERAGE VALUE*, ALSO ALL THE VALUES CONVERTED *KILOGRAMS***



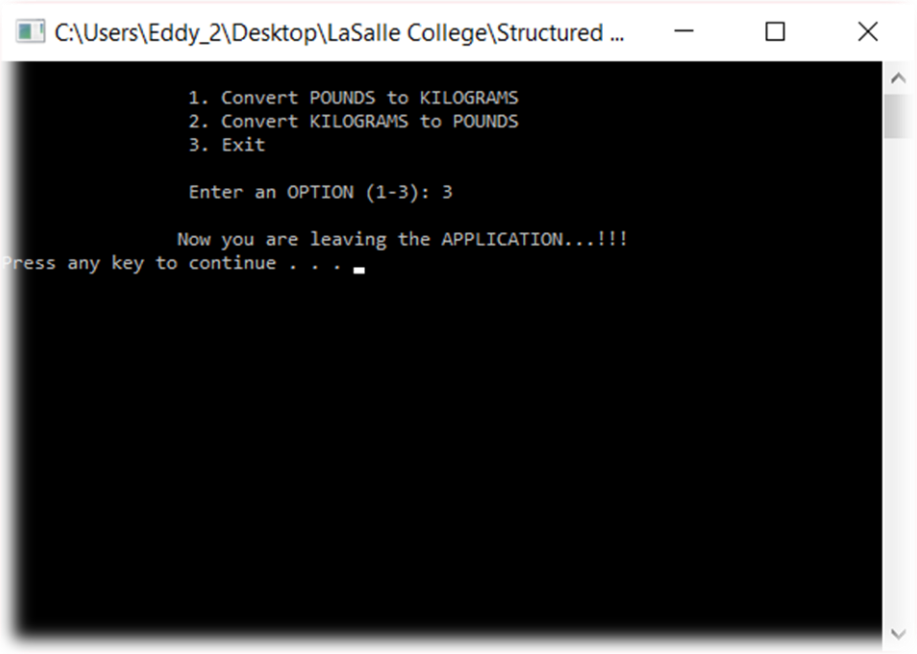

***OPTION 2:* CONVERT KILOGRAMS TO POUNDS**




**SHOW *MAXIMUM, MINIMUN AND AVERAGE VALUE*, ALSO ALL THE VALUES CONVERTED *POUNDS***




***OPTION 3:*** **EXIT**



**QUESTION 3-1: CREATE A STRUCTURE**

*Given the following data structure, declare a structure named Courses with the appropriate fields.*

| ***Course Number*** | ***Title*** | ***Hours per week*** | ***Session*** |
|---|---|---|---|
| *420-P16-AS* | *Structured Programming* | *6* | *Fall 2016* |

```
struct Session{
	string season;
	int year;
};

struct Courses{
	string courseNumber;
	string titleC;
	int hoursC;
	Session termC;
};
```

**QUESTION 3-2: RUN AND DISPLAY THE OUTPUT**

*Write a code program in C++ that allows entering all the courses data that you have for current session at compile-time using the structure from previous point. Present the output you obtain in the documentation of the project (print-screens) or by copy-paste as comment in C++ file.*

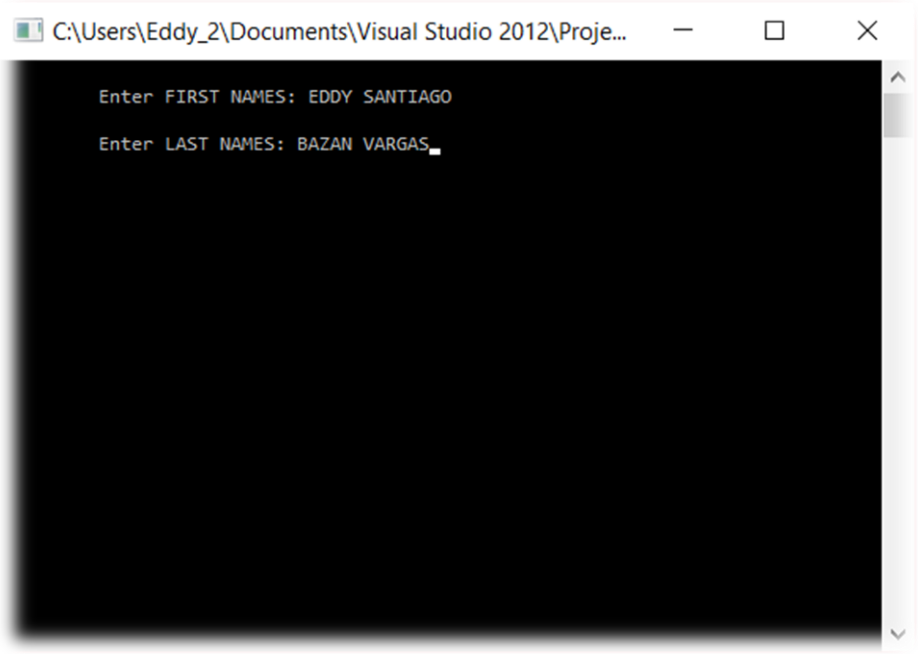
| STRUCTURE | DESCRIPTION |
|---|---|
| `struct Phone{`<br>`int code;`<br>`int area;`<br>`int part1;`<br>`int part2;`<br>`};` | Structure type called *Phone* with parameters integer *code, area, part1, part2* |
| `struct Person{`<br>`int numberCode;`<br>`string firstName;`<br>`string lastName;`<br>`string email;`<br>`Phone phoneNumber;`<br>`};` | Structure type called *Person* with parameters integer *numberCode,* string *firstName, lastName, email* and structure Phone *phoneNumber* |

| VARIABLES | DESCRIPTION |
|---|---|
| `int i, codeNumber, optC;` | Global variables type INTEGER |
| `vector <Person> table(0);` | Vector type PERSON (Structure) |

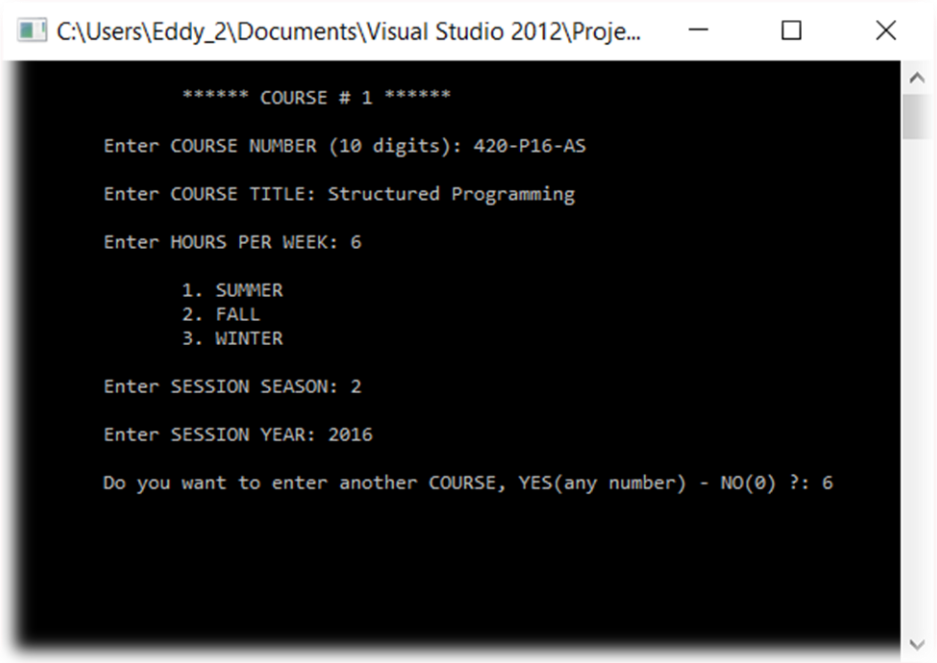
| | |
|---|---|
| `Person data;` | Variable type PERSON (Structure) |
| `Phone test;` | Variable type Phone (Structure) |

| FUNCTION NAME | DESCRIPTION |
|---|---|
| `void CreateAddStudent(vector <Person>& table, Person data);` | Create and add a new student to the vector |
| `void SearchStudent(vector <Person>& table);` | Search a student with his CODE NUMBER in the vector |
| `void DisplayStudent(vector <Person>& table);` | Show all the students in the vector |
| `int validateCodeNumber(vector <Person>& table, int value);` | Return 0 if the code number doesn't exist in the vector.<br>Return 1 if the code number exists in the vector. |
| `Phone insertPhone();` | Return a value type Phone that the user insert the phone number |
| `Phone AreaAndNumber();` | Return a value type Phone with the code area and the the last 7 digits of the number that the user enter |

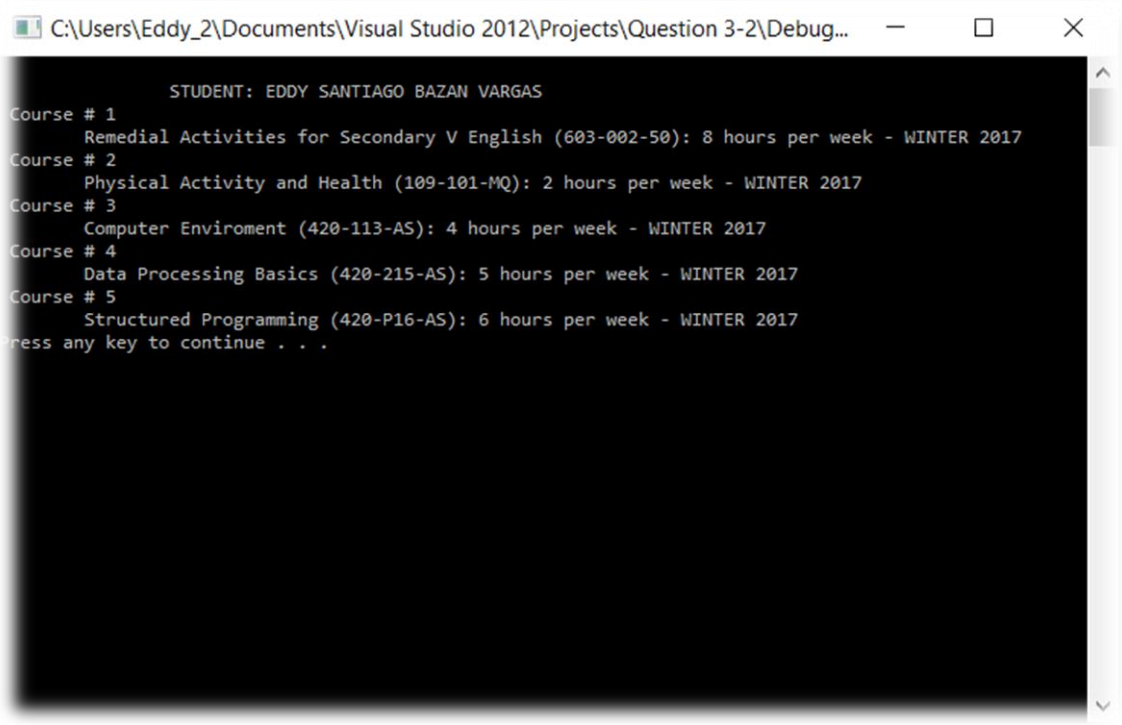
## ENTER FIRST AND LAST NAMES FOR A STUDENT




## ENTER CORUSE INFORMATION



## SHOW ALL COURSES THAT THE STUDENT HAVE IN THE SEMESTER




## SECTION II: Programming in C++

*Create an application that keeps track of students, knowing that a student is defined by: a number, a first name, a last name, an email and a phone number where first name and last name are defined as a structure Person and phone number is a structure Phone with the following properties:*

*1- international code : 1digit*

*2- country code : 2 digits*

*3- regional code : 3 digit*

*4- home code : 7 digits*

*The menu is as follows:*

***Student Management Application***

*1. Create and add a student.*

*2. Search for a student by number*

*3. Display the student list*

*4 Exit the application*

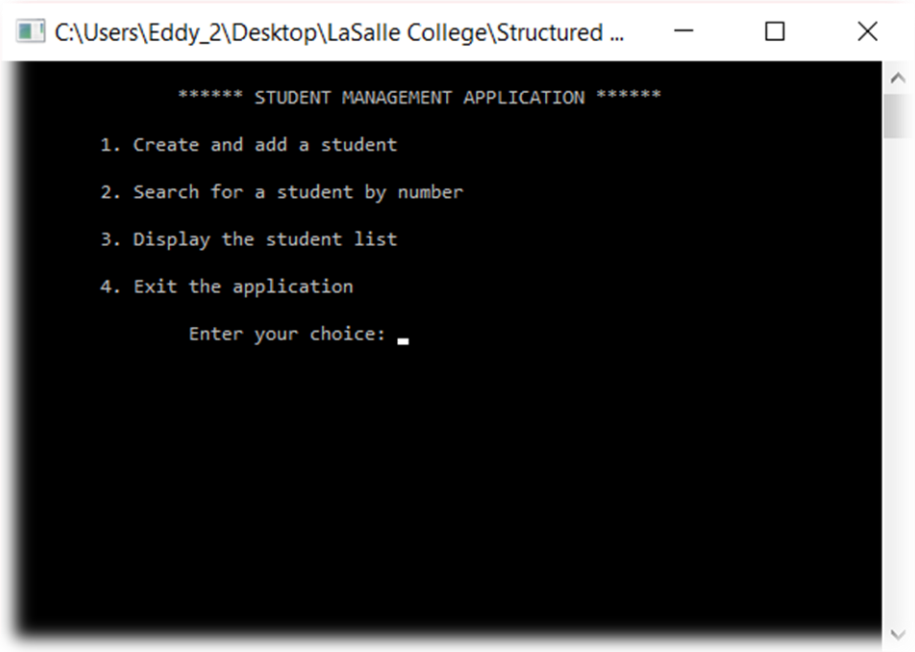
*Enter your choice: _*

| VARIABLES | DESCRIPTION |
|---|---|
| `int i, opt;` | Global variables type INTEGER |

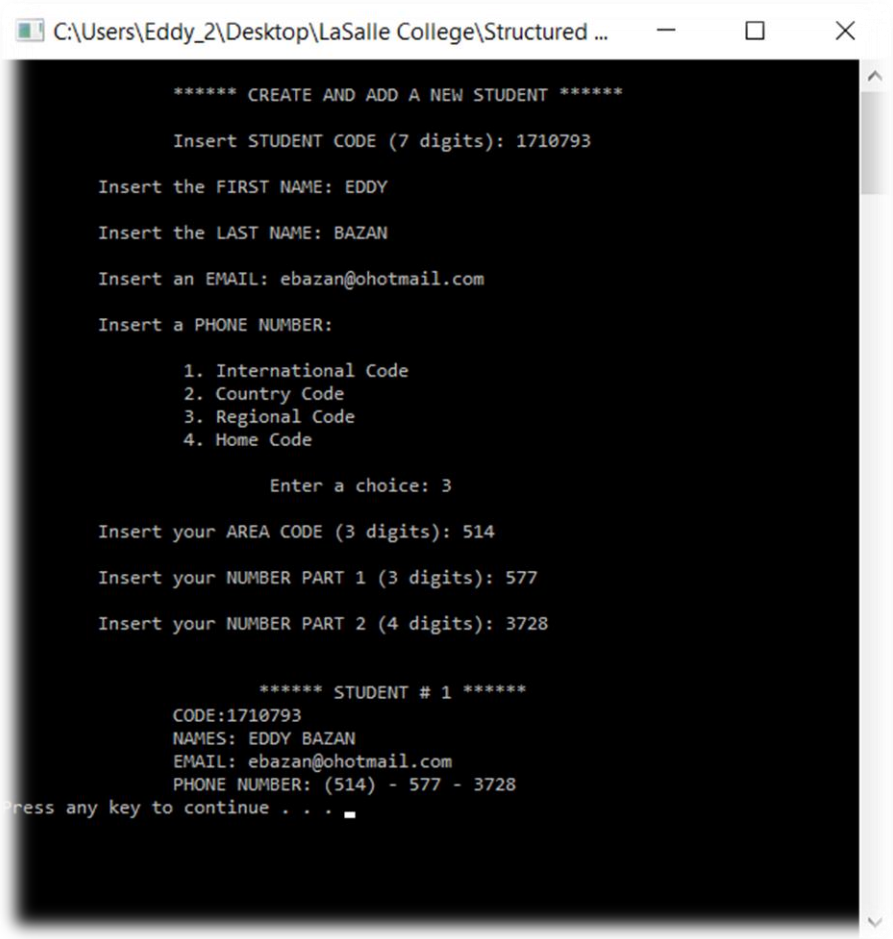
| | |
|---|---|
| `const int SIZE=10;` | Constant INTEGER with value 15 |
| `string FirstNames, LastNames, seasonS;` | Global variables type STRING |
| `vector <Courses> tableS(0);` | Vector type Courses (Structure) |

| FUNCTION NAME | DESCRIPTION |
|---|---|
| `string insertCourseNumber();` | Return a value *String* with the value of de Course Number that the user enter |
| `Session term();` | Return a value *Session* (Structure) with the value of de term that the user enter |
| `void displayCourses(vector <Courses>& table);` | Show all the courses that the user enter |

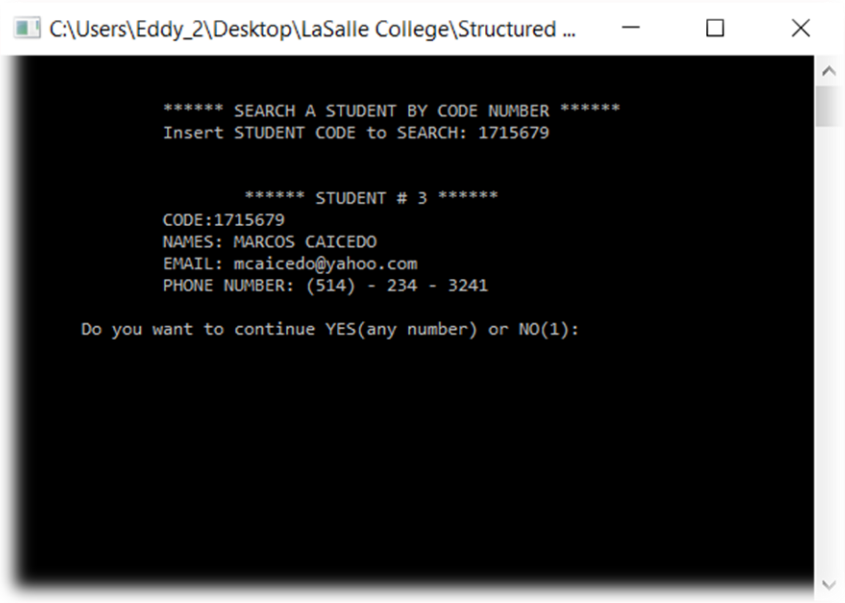
**QUESTION 4: DISPLAY THE STUDENT MANAGEMENT MENU**



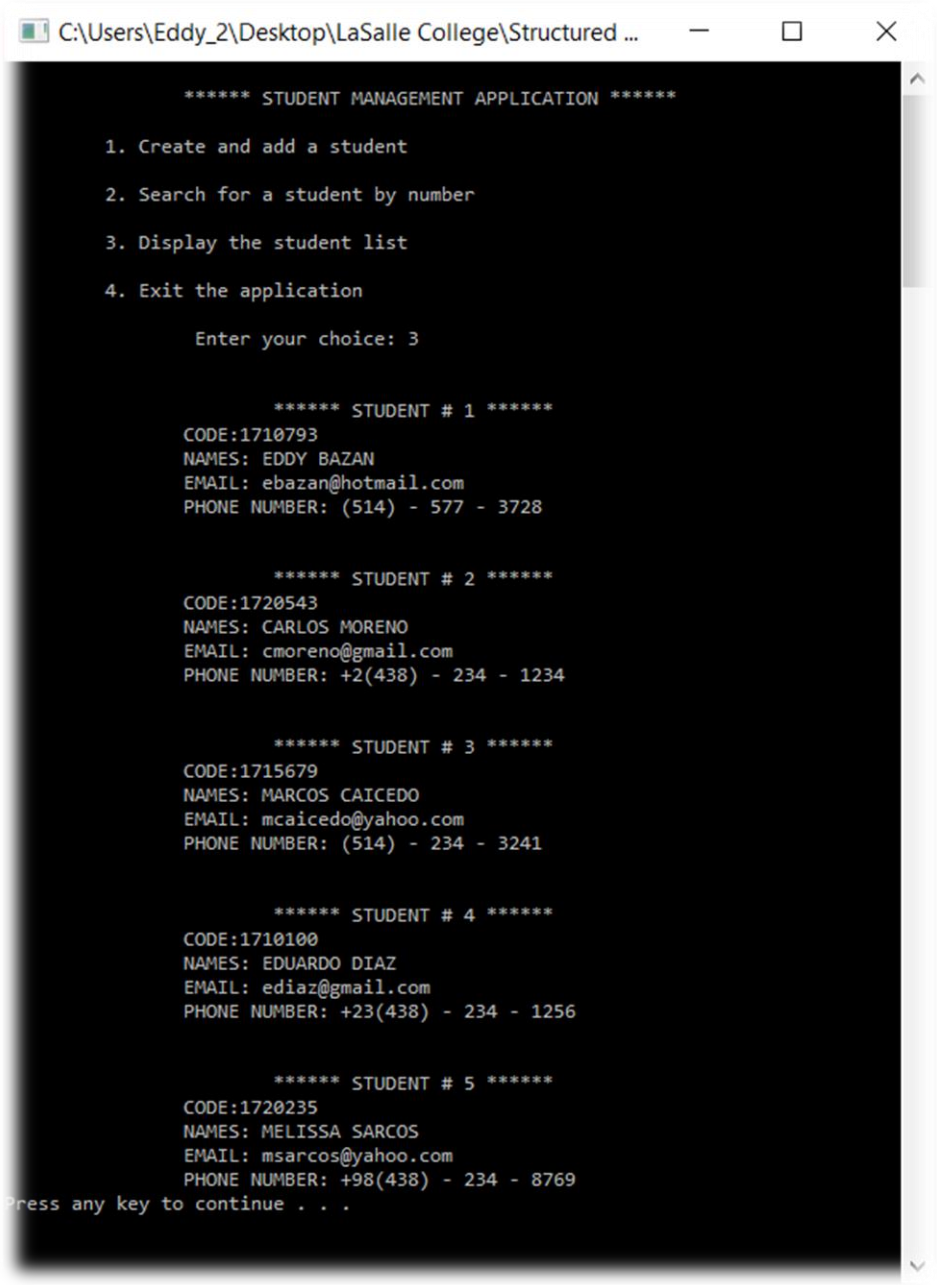
## QUESTION 5: INPUT STUDENT INFORMATION USING STRUCTURE




## QUESTION 6: SEARCH FOR STUDENT BY STUDENT NUMBER



**QUESTION 7: DISPLAY STUDENT LIST**



**QUESTION 8: EXIT THE APPLICATION**