

Body Mass index (BMI)

Advanced Computational Methods Project

Name: Mamoun Al-Mardini

ID: @44332

Abstract:

Obesity is one of the most public health problems affecting the society, and physicians are interested in finding the categories that an individual falls into to determine if he/she is underweight, normal or overweight. For this reason, a formula has been developed by Belgian polymath Adolphe Quetelet during the course of developing "social physics" and called Body Mass Index (BMI).

Introduction:

Body mass index is defined as the individual's body mass divided by the square of his or her height. The formulae universally used in medicine produce a unit of measure of kg/m^2 . In order to calculate the BMI for a person, the height and weight must be provided, so in this project I will simulate the values by using two sensors, one for height and another for weight, and these sensors will provide analog(voltage) values.

The weight sensor can measure maximum weight of 200 Kg, and each 2.5 mV will represent 0.1 Kg, on the other hand, the height sensor can measure maximum height of 250 cm, and each 20 mV will represent 1 cm.

BMI categories:

The following figure shows the BMI categories that an individual may fall into based on his/her height and weight:



Figure.1

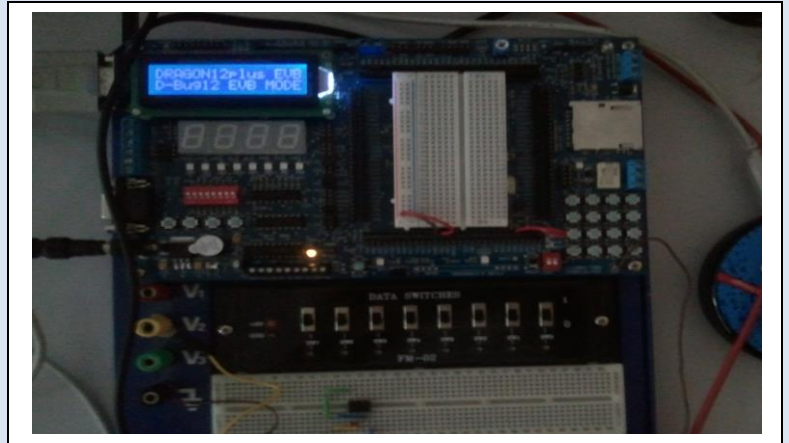
Table.1 includes the exact range for each category:

BMI	Category
18.5 or Less	Underweight
18.5-24.9	Normal
25-29.9	Overweight
30-34.9	Obese 1
35-39.9	Obese 2
40 or greater	Extremely Obese

Table.1

Components of the project:

- Motorola Microcontroller (HCS12/9S12).
- Weight cell sensor with a resolution of 2.5 mV/0.1 kg, measuring a maximum weight of 200 kg.
- Height cell sensor with resolution 20 mV/1 cm, measuring a maximum height of 250 cm.
- Weight Cell sensor is connected to ATD-CH0.
- Height cell sensor is connected to ATD-CH1.



Theoretical Derivation

- Sensors are available to measure all sorts of physical quantities and many of these sensors provide DC signals in the 0 to 5 volt range
- Some sensors generate an output voltage in the range of $0 \sim V_Z$, where $V_Z < V_{DD}$ (5 volts).
- V_Z can be much smaller than V_{DD} .
- When V_Z is much smaller than V_{DD} , the A/D conversion result cannot be accurate.
- The solution to this problem is to use an scaling circuit to amplify the transducer output to cover the whole range of 0 V to V_{DD} .
- Weight cell sensor with a resolution of 2.5 mV/0.1 kg is used in this project.
- Height cell sensor with resolution 20 mV/1 cm is used in this project.
- The value generated from the analog to digital conversion is multiplied by 0.0025 for height, and by 0.2 for weight, as shown in the following formulas:

```
height = ATDConversion*0.0025  
weight = ATDConversion *0.2;
```

- Height and Weight found the previous two equations are then inserted in the following formula to find the BMI value:

$$\text{BMI} = \text{weight} / (\text{Height} * \text{height})$$

Numerical Formulation

Since two points (weight and height) are used to generate the BMI function, **Bilinear interpolation** is needed to find the approximate BMI value for the given weight and height values.

The formula used to find the interpolation for two points is in the following form:

$$P \cong \frac{(X_2 - X)(Y_2 - Y)}{(x_2 - x_1)(Y_2 - Y_1)} Q_{11} + \frac{(X - X_1)(Y_2 - Y)}{(x_2 - x_1)(Y_2 - Y_1)} Q_{21} + \frac{(X_2 - X)(Y - Y_1)}{(x_2 - x_1)(Y_2 - Y_1)} Q_{12} + \frac{(X - X_1)(Y - Y_1)}{(x_2 - x_1)(Y_2 - Y_1)} Q_{22}$$

Where

$$X_1 = H_1$$

$$X_2 = H_2$$

$$Y_1 = W_1$$

$$Y_2 = W_2$$

X and Y are the values of height and weight to be interpolated.

$$Q_{11} = \text{BMI}(H_1 W_1)$$

$$Q_{12} = \text{BMI}(H_1 W_2)$$

$$Q_{21} = \text{BMI}(H_2 W_1)$$

$$Q_{22} = \text{BMI}(H_2 W_2)$$

P : Represents the BMI value after interpolation.

Figure.2 illustrate how the bilinear interpolation works; we need to find four points (Q_{11} , Q_{12} , Q_{21} , Q_{22}) in which they represent the BMI values for the upper and lower values of the height and weight near the new value.

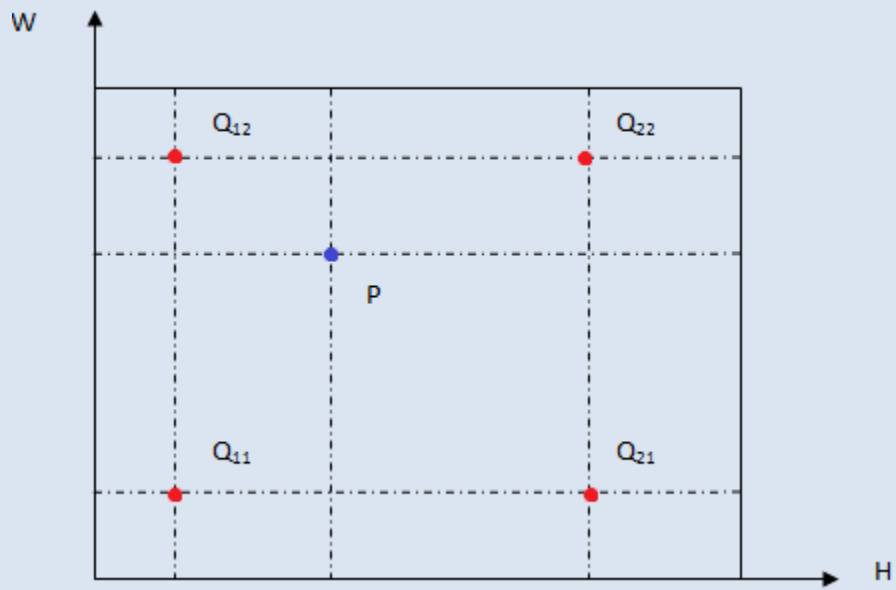
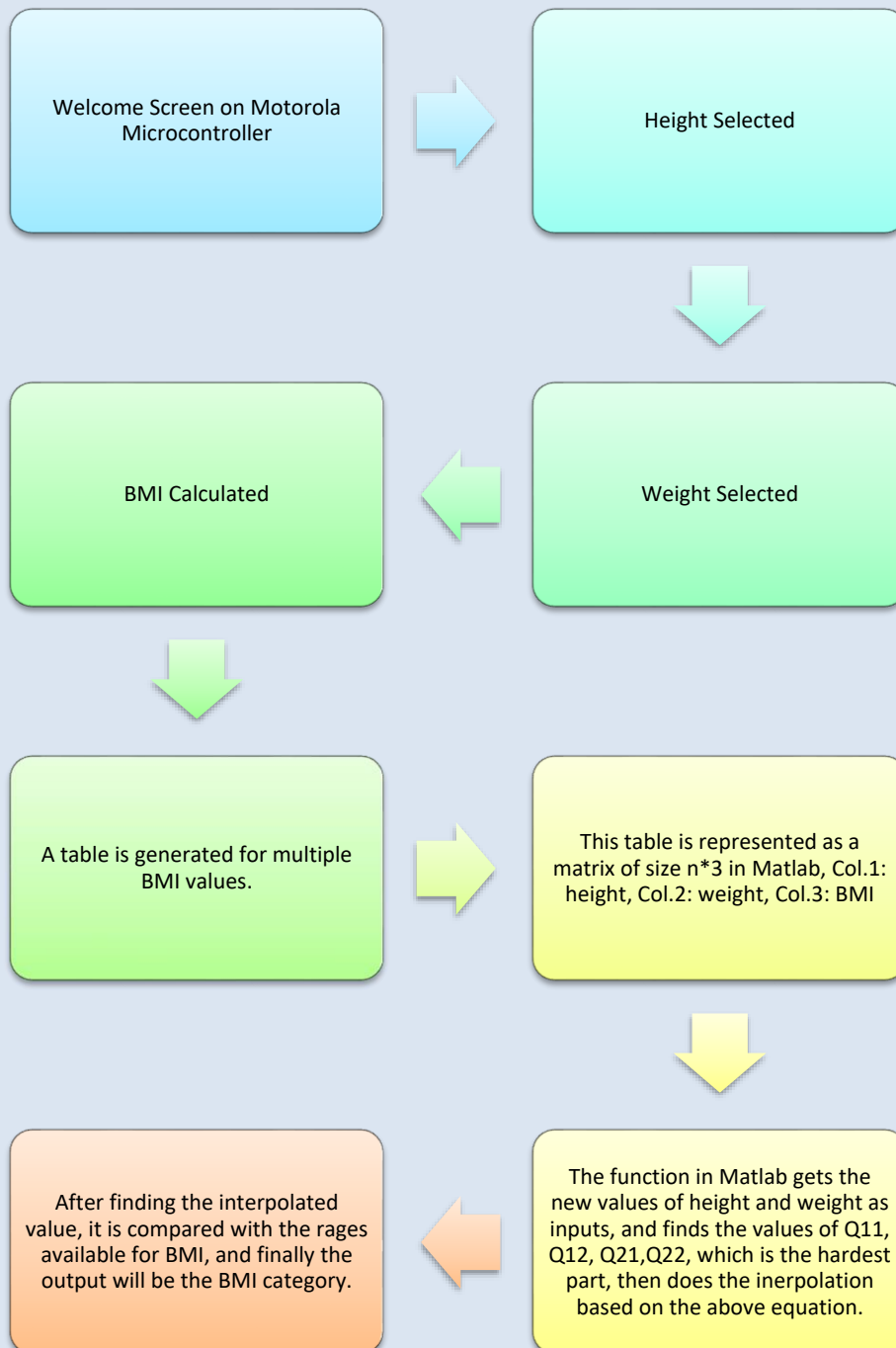


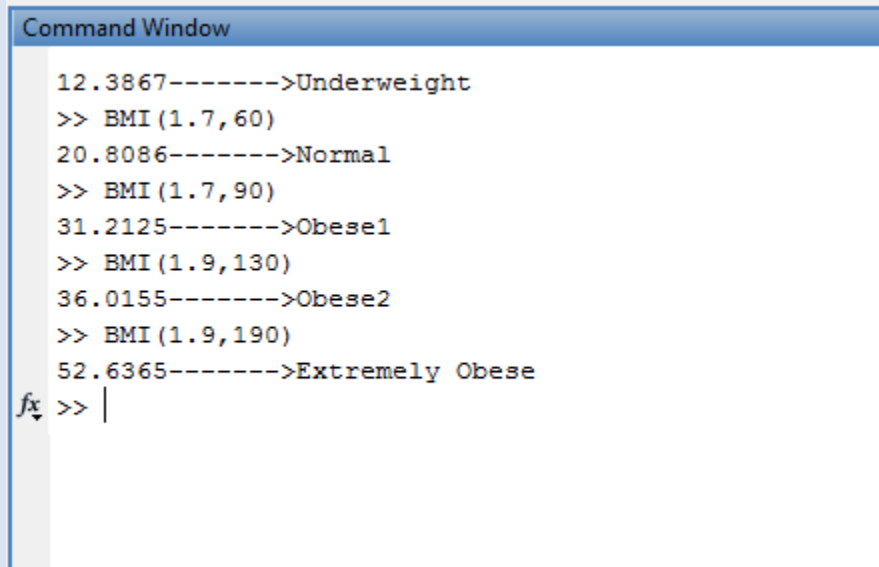
Figure.2

Flow chart:



Experimental results

Output from Matlab function :



```
Command Window
12.3867----->Underweight
>> BMI(1.7,60)
20.8086----->Normal
>> BMI(1.7,90)
31.2125----->Obese1
>> BMI(1.9,130)
36.0155----->Obese2
>> BMI(1.9,190)
52.6365----->Extremely Obese
fx >> |
```

Figure.3

Table.2 shows the extrapolated BMI value for some selected height and weight values, associated with the exact value and error:

Note: the step size for the height is 0.1 m, and for weight is 20 Kg:

A	B	C	D	E
Height	Weight	BMI	Exact value	Error
1.8	40	12.3867	12.3456	-0.0411
1.7	60	20.8086	20.7612	-0.0474
1.7	90	31.2125	31.1419	-0.0706
1.9	130	36.0155	36.0111	-0.0044
1.9	190	52.6365	52.6316	-0.0049

Table.2

Figure.4 gives a 3D graph for the interpolated and exact value of the BMI:

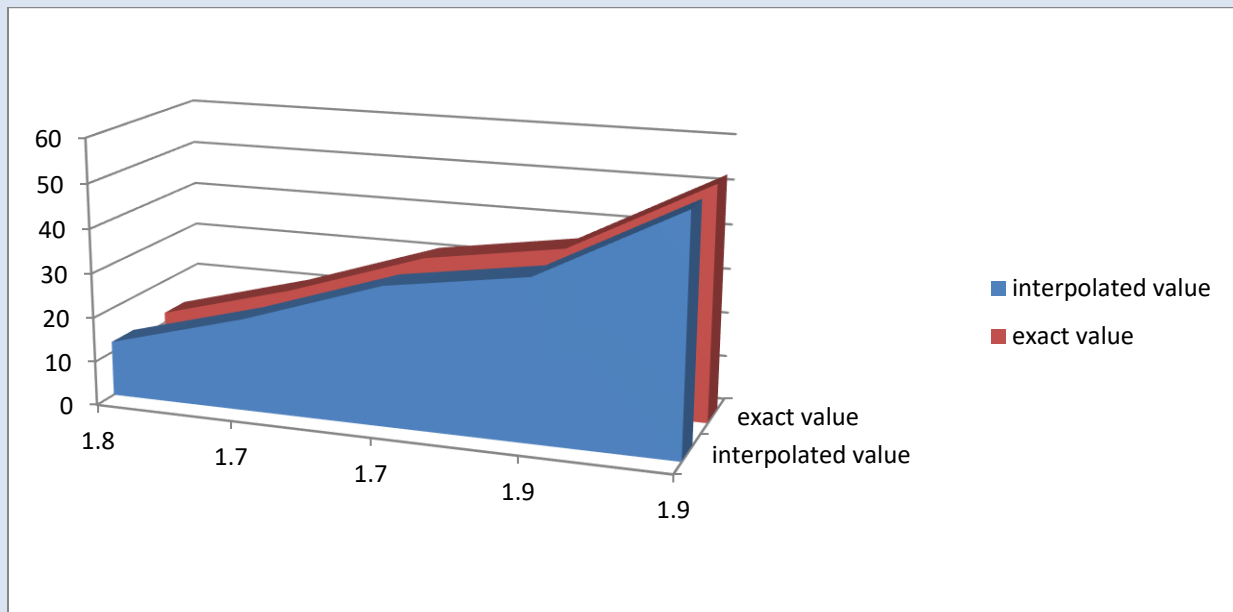


Figure.4

Discussion of results

Running the program on the Motorola Microcontroller allowed us to get the weight and height from the sensors and convert the values to digital ones in order to get the exact BMI values.

Bilinear interpolation located the values of height and values within the exact values, and found the interpolated BMI value as seen in figure.3 and as tabulated in table.2.

The exact and interpolated values are very close, and the error is very small as seen in table.2, and as illustrated in graph.4

This small error could also be decreased by decreasing the step sizes of height and weight.

Conclusions and Recommendations

In this project, we were able to use numerical methods to get a very close value for BMI without knowing the exact function, and by only using set of values that are measured previously. The accuracy of the system could be increased by decreasing the step sizes used for height and weight.

In the class, we studied how to do interpolation for one dimension and in this project, I got benefited by the class notes and did some research to understand how we can build a formula for two dimension system, so that improved my numerical knowledge, and allowed me to use numerical method in real life application.

References:

1. http://www.amazinglivingtools.com/WeightLoss/health_watch.htm
2. www.wikipedia.com
3. Han-Way Huang, HCS12/9S12 An Introduction to software and hardware interfacing 2nd edition;
4. Hoffman, Numerical Methods for Engineers and Scientists
5. <http://www.mathworks.com>
6. http://www.ajdesigner.com/phpinterpolation/bilinear_interpolation_equation.php
7. Prof. Al-Ali slides notes for embedded system class.

Appendix:

Motorolla Microcontroller Code:

```
#include <D12_LCD.h>
void delay_halfsec(void)
{
    int i;
    for(i=0; i<50000; i++)delay_10usec();
}
void delay_100usec(void)
{
    int i;
    for(i=0; i<10; i++)delay_10usec();
}
void initialize_ADC(void)
{
    ATD0CTL2 = 0xC0; // power up ADC, disable interrupts
    delay_100usec(); // wait for ADC to warm up
    ATD0CTL3 = 0x08; // select active background mode 1 conv
    ATD0CTL4 = 0x2B; // sample time 2 ADC clock, prescale of 11, 10-bit
}
float ADC_CONVERT_W(void)
{
    char *Voutbuf;
    int *status;
    float weight;
    clr_disp();
    while((PTH&0X04) != 0X04){
        lcd_print_txt("Weight (Kg)= ", LINE1);
        ATD0CTL5 = (0x86); // channel no. and justification
        while(!(ATD0STAT0 & 0x80)); // wait for conversion to finish
        weight = ATD0DR0*0.2;
        if(weight>200){
            weight = 200;
        }
        else if(weight<=0){
            weight = 10;
        }
        Voutbuf = ftoa(weight, status);
        lcd_print_txt(Voutbuf, LINE2);
        put_char (' ');
        put_char ('K');
        put_char ('g');
        delay_halfsec();
        clr_disp();
        lcd_print_txt("Are you ready?", LINE1);
        lcd_print_txt("Put DP3 to high", LINE2);
        delay_halfsec();
        clr_disp();
    }
    return(weight); // get and return the value to the caller
}

float ADC_CONVERT_H(void)
```

```

{
    char *Voutbuf;
    int *status;
    float height;
    clr_disp();
    while((PTH&0X02) != 0X02){
        lcd_print_txt("Height (M)= ", LINE1);
        ATD0CTL5 = (0x87); // channel no. and justification
        while(!(ATD0STAT0 & 0x80)); // wait for conversion to finish
        height = ATD0DR0*0.0025;
        if(height > 2.5){
            height = 2.5;
        }
        else if(height <= 0){
            height = 0.1;
        }
        Voutbuf = ftoa(height, status);
        lcd_print_txt(Voutbuf, LINE2);
        put_char (' ');
        put_char ('M');
        delay_halfsec();
        clr_disp();
        lcd_print_txt("Are you ready?", LINE1);
        lcd_print_txt("Put DP2 to high", LINE2);
        delay_halfsec();
        clr_disp();
    }
    return(height); // get and return the value to the caller
}

void welcome_screen(void){
    clr_disp();
    lcd_print_txt("Welcome to the", LINE1);
    lcd_print_txt("BMI calculator", LINE2);
    delay_halfsec();
    clr_disp();
    lcd_print_txt("Ready to start ?", LINE1);
    lcd_print_txt("Put DP1 to high", LINE2);
    while((PTH&0X01) != 0X01);
}

void BMI_CHECK_DISP(float BMI, float weight, float height){
    char *Voutbuf;
    int *status;
    clr_disp();

    while((PTH&0X08) != 0X08){
        Voutbuf = ftoa(weight, status);
        lcd_print_txt(Voutbuf, LINE2);
        delay_halfsec();
        clr_disp();
        Voutbuf = ftoa(height, status);
        lcd_print_txt(Voutbuf, LINE2);
        delay_halfsec();
        clr_disp();
        Voutbuf = ftoa(BMI, status);
        lcd_print_txt(Voutbuf, LINE2);
        delay_halfsec();
        clr_disp();
    }
}

```

```

        if(BMI<=16.5){

            PORTB = 0x01;
        }
        else if(BMI>16.5 && BMI<=18.5){

PORTB = 0x02;
        }
        else if(BMI>18.5 && BMI<=25.0){

PORTB = 0x04;
        }
        else if(BMI>25.0 && BMI<=30.0){

PORTB = 0x08;
        }
        else if(BMI>30.0 && BMI<=35.0){

PORTB = 0x10;
        }
        else if(BMI>35.0 && BMI<=40.0){

PORTB = 0x20;
        }
        else if(BMI>40){

PORTB = 0x40;
        }
        delay_halfsec();
        clr_disp();
    }

    delay_halfsec();
    clr_disp();
    }
    PORTB = 0x00;
}

float BMI_CALCULATE(float height, float weight){
    float BMI_value;
    BMI_value = weight/(height*height);
    return (BMI_value);
}

void init_ports(void){
    DDRB = 0xFF;
    DDRJ = 0xFF;
    DDRP = 0xFF;
    DDRH = 0x00;
    PTJ = 0x00;
    PTP = 0xFF;
}

void main()
{
    float height;
    float weight;
    float bmi_value;
    init_ports();
    init_lcd();

```

```

initialize_ADC();

while(1)
{
    welcome_screen();
    height = ADC_CONVERT_H();
    weight = ADC_CONVERT_W();
    bmi_value = BMI_CALCULATE(height, weight);
    BMI_CHECK_DISP(bmi_value, weight, height);

}
}

```

Matlab Code:

```

function [] = BMI(height,weight)
%A is a matrix with size of n*3, the first column represents the height,
%second column represents the weight, and the third one represents BMI
%value. these values are calculated using the sensors and Motorola
%microcontriller.

%height and weight variables are the height and weight respectively to be
interpolated
Q11=0;
Q12=0;
Q21=0;
Q22=0;

Q11_index=0;
Q12_index=0;
Q21_index=0;
Q22_index=0;

A=[
1.0012 40.08 39.9837;
1.0012 60.119 59.975;
1.0012 80.158 79.9659;
1.0012 100.2 99.9563;
1.0012 120.23 119.9464;
1.0012 140.27 139.9361;
1.0012 160.31 159.9253;
1.0012 180.35 179.9142;
1.0012 200.38 199.9027;
1.1243 40.039 31.6753;
1.1243 60.058 47.5125;
1.1243 80.077 63.3494;
1.1243 100.1 79.186;
1.1243 120.11 95.0222;
1.1243 140.13 110.8582;
1.1243 160.15 126.6938;
1.1243 180.16 142.5291;
1.1243 200.18 158.3641;

```

1.2022 40.04 27.7036;
1.2022 60.059 41.555;
1.2022 80.078 55.4062;
1.2022 100.1 69.257;
1.2022 120.11 83.1076;
1.2022 140.13 96.9579;
1.2022 160.15 110.8079;
1.2022 180.17 124.6576;
1.2022 200.18 138.5071;
1.3125 40.12 23.2896;
1.3125 60.179 34.934;
1.3125 80.238 46.5782;
1.3125 100.3 58.2222;
1.3125 120.36 69.866;
1.3125 140.41 81.5095;
1.3125 160.47 93.1528;
1.3125 180.53 104.7958;
1.3125 200.58 116.4386;
1.3998 40.448 20.6427;
1.3998 60.672 30.9638;
1.3998 80.895 41.2846;
1.3998 101.12 51.6052;
1.3998 121.34 61.9256;
1.3998 141.56 72.2459;
1.3998 161.78 82.5659;
1.3998 182 92.8857;
1.3998 202.22 103.2052;
1.5012 40.084 17.7865;
1.5012 60.125 26.6795;
1.5012 80.166 35.5724;
1.5012 100.21 44.465;
1.5012 120.25 53.3575;
1.5012 140.29 62.2498;
1.5012 160.33 71.1419;
1.5012 180.36 80.0338;
1.5012 200.4 88.9256;
1.6112 40.048 15.4269;
1.6112 60.071 23.1401;
1.6112 80.094 30.8531;
1.6112 100.12 38.5661;
1.6112 120.14 46.2788;
1.6112 140.16 53.9914;
1.6112 160.18 61.7038;
1.6112 180.2 69.4161;
1.6112 200.22 77.1282;
1.7001 40.091 13.8708;
1.7001 60.137 20.806;
1.7001 80.181 27.7411;
1.7001 100.23 34.676;
1.7001 120.27 41.6108;
1.7001 140.31 48.5455;
1.7001 160.36 55.48;
1.7001 180.4 62.4143;
1.7001 200.44 69.3486;
1.8122 40.084 12.2055;
1.8122 60.125 18.3081;
1.8122 80.166 24.4106;

1.8122 100.21 30.5129;
1.8122 120.25 36.6151;
1.8122 140.29 42.7172;
1.8122 160.33 48.8192;
1.8122 180.36 54.921;
1.8122 200.4 61.0227;
1.8991 40.004 11.0918;
1.8991 60.005 16.6376;
1.8991 80.006 22.1832;
1.8991 100.01 27.7288;
1.8991 120.01 33.2742;
1.8991 140.01 38.8195;
1.8991 160 44.3647;
1.8991 180 49.9098;
1.8991 200 55.4547;
2.001 40.043 10.0008;
2.001 60.064 15.0011;
2.001 80.085 20.0012;
2.001 100.11 25.0013;
2.001 120.12 30.0012;
2.001 140.14 35.0011;
2.001 160.16 40.0008;
2.001 180.18 45.0005;
2.001 200.2 50;
2.1 40.052 9.082;
2.1 60.077 13.6229;
2.1 80.102 18.1637;
2.1 100.13 22.7043;
2.1 120.15 27.2449;
2.1 140.17 31.7854;
2.1 160.2 36.3259;
2.1 180.22 40.8662;
2.1 200.24 45.4064;
2.207 40.012 8.2145;
2.207 60.017 12.3216;
2.207 80.022 16.4287;
2.207 100.03 20.5356;
2.207 120.03 24.6425;
2.207 140.03 28.7493;
2.207 160.04 32.856;
2.207 180.04 36.9627;
2.207 200.04 41.0692;
2.303 40.457 7.6279;
2.303 60.685 11.4417;
2.303 80.912 15.2554;
2.303 101.14 19.0691;
2.303 121.37 22.8827;
2.303 141.59 26.6962;
2.303 161.82 30.5096;
2.303 182.04 34.323;
2.303 202.27 38.1363;
2.399 40.452 7.0288;
2.399 60.678 10.5431;
2.399 80.903 14.0574;
2.399 101.13 17.5715;
2.399 121.35 21.0856;
2.399 141.58 24.5996;


```

2.399 161.8 28.1136;
2.399 182.02 31.6274;
2.399 202.24 35.1412;
2.5002 40.449 6.4707;
2.5002 60.672 9.706;
2.5002 80.896 12.9412;
2.5002 101.12 16.1764;
2.5002 121.34 19.4114;
2.5002 141.56 22.6464;
2.5002 161.78 25.8814;
2.5002 182.01 29.1163;
2.5002 202.23 32.3511];

%first we need to locate the values of weight and height between two values
%by traversing matrix A

for i=1:1:144

    if A(i,1)>=height
        height_upper_index=i;
        break;
    end
end
height_lower_index=height_upper_index-9;
upperbound=height_upper_index-1;

for j=height_upper_index:1:(height_upper_index+8)
    if A(j,2)>=weight
        Q22_index=j;
        Q21_index=j-1;
        break;
    end
end
Q22=A(Q22_index,3);
Q21=A(Q21_index,3);
H2=A(Q22_index,1);
W2=A(Q22_index,2);
W2=round(W2);
W1=A(Q21_index,2);
W1=round(W1);

for k=height_lower_index:1:upperbound

    test=round(A(k,2));
    if test==W1
        Q11_index=k;
    end
    if test==W2
        Q12_index=k;
    end
end
Q11=A(Q11_index,3);
Q12=A(Q12_index,3);
H1=A(Q11_index,1);
p= ((H2-height)*(W2-weight)*Q11+(height-H1)*(W2-weight)*Q21+(H2-
height)*(weight-W1)*Q12+(height-H1)*(weight-W1)*Q22)/((H2-H1)*(W2-W1));

```

```

if p<=18.5
str=[num2str(p), '----->', 'Underweight'];
end
if 18.5<p && p<=25
    str=[num2str(p), '----->', 'Normal'];
end
if 25<p && p<=30
    str=[num2str(p), '----->', 'OverWeight'];
end
    if 30<p && p<=35
str=[num2str(p), '----->', 'Obese1'];
    end
if 35<p && p<=40
str=[num2str(p), '----->', 'Obese2'];
end
if p>40
str=[num2str(p), '----->', 'Extremely Obese'];
end
disp(str)
end

```