| **Computing Fundamentals & Programming**  ***Section*: BSCE2021 Assignment # 6 *Total marks*: 100**  ***Name*** :***Muhammad Abubakar Saif Roll number* : BSCE21017** |
| --- |

***Submission:***

Email instructor or TA in case you are facing any difficulty in writing an algorithm of any question.

• You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other.

• Plagiarism will lead to a straight zero in all previous assignments with additional consequences as well.

• Code should be yours, not from your internet or any other source.

If any instance is found copied from an online source, it will be also considered as cheating.

• Submission after due time will not be accepted.

• From now onward, heavy penalty will be there for unindented code and codes with improper comments.

• Display appropriate input and output messages whether mentioned in question or not.

• Dry run your solution. It helps in finding out bugs in your program.

## REPORT

Solve the following and comment on every step of your algorithm as if you are explaining your code to your fellows to demonstrate clear understanding. Submit the PDF of the report containing code and screenshot of output.

**Table of Content:**

[**REPORT**](#_gjdgxs) **1**

[**TASK GIVEN BY INSTRUCTORS:**](#_30j0zll) **2**

[Note (by Student):](#_whtxhupr73j3) 3

[TASK 1 [20 marks]](#_1t3h5sf) 3

[Pseudocode:](#_jahpcpwrw97p) 4

[Solid Right Angle Triangle:](#_phgptm61ddsl) 4

[Hollow right triangle:](#_413ourn0nuqa) 4

[Solid Isosceles Triangle:](#_wt0pr3pu3t57) 5

[Hollow Isosceles Triangle:](#_l05lrvsuiix8) 5

[Solid Equilateral Triangle:](#_v8l3zcjchzku) 6

[Hollow Equilateral Triangle:](#_xzkum63esvez) 6

[TASK 2 [80 marks]](#_4d34og8) 7

[Code (Function.cpp):](#_cojtwmg7rfs3) 8

[Solid Right Triangle Code:](#_8n5ingz2g7u) 8

[Hollow Right Triangle Code:](#_252wxvx4e5t3) 8

[Solid Isosceles Triangle Code:](#_w58n4f72ahc0) 9

[Hollow Isosceles Triangle Code:](#_yde0zgdm4tt) 9

[Solid Equilateral Triangle Code:](#_54irbsqgz8dn) 10

[Hollow Equilateral Triangle Code:](#_kvdwxxatpemy) 11

[Menu for Solid Shapes:](#_o0jq0sk9uw21) 12

[Menu for Hollow Shapes:](#_uhtex1npx0lh) 12

[Code (Main.cpp):](#_8vsfra3etk68) 13

[Code (Function.h):](#_93n21aka6zul) 15

[Outputs:](#_k0l977cx85jx) 16

[Solid Right Triangle:](#_9kk37761rqil) 16

[Hollow Right Triangle:](#_ysbgpd28zxx1) 16

[Solid Isosceles Triangle:](#_3jgl3scv4rwe) 17

[Hollow Isosceles Triangle:](#_jgqqjuepsjug) 17

[Solid Equilateral Triangle:](#_3ya88zkfxqfc) 18

[Hollow Equilateral Triangle:](#_jxrmltzc3alp) 18

[Note:](#_l285tnxrymnm) 18

[Errors Possibilities by Users:](#_i6tws0yd9dz1) 19

[Possibility 1:](#_76iufpffmsy4) 19

[Possibility 2:](#_yttdbmpza0p8) 19

[Possibility 3:](#_4pxhyvw884xd) 19

[Other Possibilities for User:](#_f4s0plmb1cwt) 20

[‘Exit’ Possibility:](#_1or94sldnefz) 20

[‘Restart’ Possibility:](#_ykfhfhn7w9ds) 20

## TASK GIVEN BY INSTRUCTORS:

Write a user-interactive menu-driven C++ program to print different solid and hollow shapes according to the user’s choice.

The following should appear on the screen when your program is executed:

**Enter 1 for solid, 2 for hollow and 3 for both:**

(program should continue asking for the valid input if the input is other than 1, 2 or 3)

Once user enters 1 or 2, user will be given further choice about the shape by giving the following options to the user:

**Enter your option for the shape type (1 to 8 or E to exit):**

**1) Right Triangle**

**2) Isosceles Triangle**

**3) Equilateral Triangle**

**4) All shapes**

**E) Exit**

(program should continue asking for the valid input if input is other than 1, 2, 3, 4 or E) When the user selects his/her choice, the program should prompt the user for the dimension (note that only one dimension will be required). e.g. the program should prompt as follows:

**Enter the dimension:**

Finally, the program should ask about the character to be used to draw shapes. e.g. it should prompt the user in the following manner:

**Enter the character type:**

Only these characters will be considered as valid characters: **\* ? % + $ # ~ @ ( )**

(program should continue asking for the valid input if input is other these characters)

Your program should then draw the shape(s) according to the user’s requirements.

The program should repeat itself unless the user wishes to stop.

#### Note (by Student):

Note: Headings Given in Codes are not part of the code. These are given only for convenience in reading.

### TASK 1 [20 marks]

Write pseudocode (for drawing shapes only)

#### Pseudocode:

##### Solid Right Angle Triangle:

Start

Input dim, letter

Let a = 1

While (i <= dim)

Let a = 1

While (a <= dim)

Print letter

a = a + 1

Jump to a new line

i = i + 1

End

##### Hollow right triangle:

Start

Input dim, letter

Let i = 1, last = 1

while (i <= dim)

let a = 1

let j = a

while (a <= i) { *//print characters inside a row*

if (i < 3) *//print characters in first two rows*

Print letter

if (i >= 3 and i ≠ dim and a == 1) *//print first character of all the rows except first two rows and last row*

Print letter

while (i >= 3 and j < i - 1 and i ≠ dim) *//creates spaces between first and last character of all the rows except first two rows and last row*

Print space

j = j + 1

while (i > 2 and i == dim and last <= dim) *//print characters in last row*

Print letter

last = last + 1

if (i >= 3 && a == i - 1 && i ≠ dim) *//print last character of all the rows except first two rows and last row*

Print letter

a = a + 1

Jump to new row *//ends current row*

i++

End

##### Solid Isosceles Triangle:

Start

Input dim, letter

let i = 1

let space = 1

while (i <= dim) *// makes new rows*

*Let space = i*

While (space < dim) *//creates spaces in a row*

Print a empty space

space++

Let space = 1

While ( space <= (2 \* i - 1)) *//prints letters after created spaces in the same row*

Print letter

space++

Jump to new line *//jump to new line*

i++

End

##### Hollow Isosceles Triangle:

Start

Input dim, letter

Let i = 1

Let space = 1

Let n = 1

while (i <= dim) *// makes new rows*

*Let* space = i

While (space < dim) *//creates spaces in a row*

Print an empty space

space++

Let space = 1

While ( space == 1 and i ≠ dim) *//prints first letter after created spaces in the same row*

Print letter

space++

Let mid = 1

While ( mid <= n and i ≠ 1 and i ≠ dim) *//creates spaces between first and last character*

Print an empty space *//of the row except for first and last row*

mid++

if (i ≠ 1 andi ≠ dim) *//print last character of the rows except for first and last row*

*Print* letter

Let last = 1

While ( last <= (2 \* i - 1) and i == dim) *//prints letters in the last row*

Print letter;

last++

*Jump to new line*

n = n + 2

if (i<2 and i>0) *//runs only once during this function to reset the value of n to 1 in case of i = 1*

n = 1 *// for proper functioning of this code*

i++

End

##### Solid Equilateral Triangle:

Start

Input dim, letter

Let i = 1

Let space = 1

Let n = 1

while (i <= dim) *// makes new rows*

space = i

While (space < dim) *//creates spaces in a row*

Print an empty space

space++

space = 1

While (space == 1 and i <= 1) *//prints first letter after created spaces in first row*

Print letter

space++

Let mid = 1

While ( mid <= n and i > 1) { *// print letters and spaces in all the rows except first row*

Print letter

Print an empty space

mid++

J*ump to new line*

n++

i++

End

##### Hollow Equilateral Triangle:

Start

Input dim, letter

Let i = 1

Let space

Let n

while (i <= dim) *// makes new rows*

*Let* space = i

While (space < dim) *//creates spaces in a row*

Create an empty space

space++

Let space = 1

While (space == 1 and i <= 1) *//prints first letter after created spaces in first row*

Print letter

Space++

Let mid = 1

While (mid <= i and i > 1 and (i < 3 or i == dim)) *// print letters and spaces in all the rows except last row*

Print letter

Print an empty space

mid++

Let space = 1

for (space == 1 and i >= 3 and i < dim) *//prints letter after created spaces in the first and last row*

Print letter

space++

space = 1

While (space <= n and i >= 3 and i ≠ dim) *//creates spaces in rows except in first two and last row*

Print an empty space

space++

Let space = 1

While (space == 1 and i >= 3 and i < dim) *//prints last letter in rows except in first two and last row*

Print letter

space++

Jump *to new line*

n = n + 2

if (i<3 and i>1) *//runs only once during this function to reset the value of n to 3 in case of i = 2*

n = 3; *// for proper functioning of this code*

i++

End

### TASK 2 [80 marks]

Write code in C++ for above given task

#### Code (Function.cpp):

#include **"Functions.h"**

#include **<iostream>**

**using namespace** std;

##### Solid Right Triangle Code:

**void** solRightTri(**int** dim, **char** letter) { *//print solid right triangle*

**int** i = 1;

**while** (i <= dim) { *//print a new row*

**int** a = 1;

**while** (a <= i) { *// print characters in row simultaneously in increasing order*

cout << letter;

a++;

}

cout << endl;

i++;

}

}

##### Hollow Right Triangle Code:

**void** holRightTri(**int** dim, **char** letter) { *//print hollow right triangle*

**int** i = 1, last = 1;

**while** (i <= dim) { *//print a new row*

**int** a = 1;

**int** j = a;

**while** (a <= i) { *//print characters inside a row*

**if** (i < 3) { *//print characters in first two rows*

cout << letter;

}

**if** (i >= 3 && i != dim && a == 1) { *//print first character of all the rows except first two rows and last row*

cout << letter;

}

**while** (i >= 3 && j < i - 1 && i != dim) { *//creates spaces between first and last character of all the rows except first two rows and last row*

cout << **" "**;

j++;

}

**while** (i > 2 && i == dim && last <= dim){ *//print characters in last row*

cout << letter;

last++;

}

**if** (i >= 3 && a == i - 1 && i != dim) { *//print last character of all the rows except first two rows and last row*

cout << letter;

}

a++;

}

cout << endl; *//ends current row*

i++;

}

}

##### Solid Isosceles Triangle Code:

**void** solIsoTri(**int** dim, **char** letter) {

**int** i = 1;

**int** space = 1;

**while** (i <= dim) { *// makes new rows*

**for** (space = i; space < dim; space++) { *//creates spaces in a row*

cout << **" "**;

}

**for** (space = 1; space <= (2 \* i - 1); space++) { *//prints letters after created spaces in the same row*

cout << letter;

}

cout << endl; *//jump to new line*

i++;

}

}

##### Hollow Isosceles Triangle Code:

**void** holIsoTri(**int** dim, **char** letter) {

**int** i = 1;

**int** space = 1;

**int** n = 1;

**while** (i <= dim) { *// makes new rows*

**for** (space = i; space < dim; space++) { *//creates spaces in a row*

cout << **" "**;

}

**for** (space = 1; space == 1 && i != dim; space++) { *//prints first letter after created spaces in the same row*

cout << letter;

}

**for** (**int** mid = 1; mid <= n && i != 1 && i != dim; mid++) { *//creates spaces between first and last character*

cout << **" "**; *//of the row except for first and last row*

}

**if** (i != 1 && i != dim) { *//print last character of the rows except for first and last row*

cout << letter;

}

**for** (**int** last = 1; last <= (2 \* i - 1) && i == dim; last++) { *//prints letters in the last row*

cout << letter;

}

cout << endl; *//jump to new line*

n = n + 2;

**if** (i<2 && i>0){ *//runs only once during this function to reset the value of n to 1 in case of i = 1*

n = 1; *// for proper functioning of this code*

}

i++;

}

}

##### Solid Equilateral Triangle Code:

**void** solEqTri(**int** dim, **char** letter) {

**int** i = 1;

**int** space = 1;

**int** n = 1;

**while** (i <= dim) { *// makes new rows*

**for** (space = i; space < dim; space++) { *//creates spaces in a row*

cout << **" "**;

}

**for** (space = 1; space == 1 && i <= 1; space++) { *//prints first letter after created spaces in first row*

cout << letter;

}

**for** (**int** mid = 1; mid <= n && i > 1; mid++) { *// print letters and spaces in all the rows except first row*

cout <<letter;

cout << **" "**;

}

cout << endl; *//jump to new line*

n++;

i++;

}

}

##### Hollow Equilateral Triangle Code:

**void** holEqTri(**int** dim, **char** letter) {

**int** i = 1;

**int** space;

**int** n;

**while** (i <= dim) { *// makes new rows*

**for** (space = i; space < dim; space++) { *//creates spaces in a row*

cout << **" "**;

}

**for** (space = 1; space == 1 && i <= 1; space++) { *//prints first letter after created spaces in first row*

cout << letter;

}

**for** (**int** mid = 1; mid <= i && i > 1 && (i < 3 || i == dim); mid++) { *// print letters and spaces in all the rows except last row*

cout << letter;

cout << **" "**;

}

**for** (space = 1; space == 1 && i >= 3 && i < dim; space++) { *//prints letter after created spaces in the first and last row*

cout << letter;

}

**for** (space = 1; space <= n && i >= 3 && i != dim; space++) { *//creates spaces in rows except in first two and last row*

cout << **" "**;

}

**for** (space = 1; space == 1 && i >= 3 && i < dim; space++) { *//prints last letter in rows except in first two and last row*

cout << letter;

}

cout << endl; *//jump to new line*

n = n + 2;

**if** (i<3 && i>1) { *//runs only once during this function to reset the value of n to 3 in case of i = 2*

n = 3; *// for proper functioning of this code*

}

i++;

}

}

##### Menu for Solid Shapes:

**void** solShapes(**int** shape, **int** dim, **char** letter) { *// switch cases for solid shapes*

cout <<**"Solid "**;

**switch** (shape) {

**case** 1:

cout << **"Right Triangle: "** << endl;

solRightTri(dim, letter);

**break**;

**case** 2:

cout << **"Isosceles Triangle: "** << endl;

solIsoTri(dim, letter);

**break**;

**case** 3:

cout << **"Equilateral Triangle: "** << endl;

solEqTri(dim, letter);

**break**;

**case** 4:

cout << **"Right Triangle: "** << endl;

solRightTri(dim, letter);

cout << **"Solid Isosceles Triangle: "** << endl;

solIsoTri(dim, letter);

cout << **"Solid Equilateral Triangle: "** << endl;

solEqTri(dim, letter);

**break**;

}

}

##### Menu for Hollow Shapes:

**void** holShapes(**int** shape, **int** dim, **char** letter) { *//switch cases for hollow shapes*

cout <<**"Hollow "**;

**switch** (shape) {

**case** 1:

cout << **"Right Triangle: "** << endl;

holRightTri(dim, letter);

**break**;

**case** 2:

cout << **"Isosceles Triangle: "** << endl;

holIsoTri(dim, letter);

**break**;

**case** 3:

cout << **"Equilateral Triangle: "** << endl;

holEqTri(dim, letter);

**break**;

**case** 4:

cout << **"Right Triangle: "** << endl;

holRightTri(dim, letter);

cout << **"Hollow Isosceles Triangle: "** << endl;

holIsoTri(dim, letter);

cout << **"Hollow Equilateral Triangle: "** << endl;

holEqTri(dim, letter);

**break**;

}

}

#### Code (Main.cpp):

#include **<iostream>**

#include **"Functions.h"**

**using namespace** std;

**int** main() {

**int** input, dim, realShape;

**char** shape;

**char** letter;

string rep;

again:

**do** {

cout << **"Enter 1 for solid, 2 for hollow and 3 for both:"** << endl;

cin >> input;

} **while** (input > 3 && cout << **"Wrong Input, Try Again:"** << endl);

**do** {

cout << **"Enter your option for the shape type (1 to 4 or E to exit): "** << endl;

cout << endl;

cout << **"1) Right Triangle"** << endl;

cout << **"2) Isosceles Triangle"** << endl;

cout << **"3) Equilateral Triangle"** << endl;

cout << **"4) All shapes"** << endl;

cout << **"E) Exit"** << endl;

cin >> shape;

} **while** (shape != **'1'** && shape != **'2'** && shape != **'3'** && shape != **'4'** && shape != **'E'** &&

cout << **"Wrong Input, Try Again:"** << endl);

**if** (shape == **'E'**) {

**return** 0;

}

**if** (shape == **'1'**) {

realShape = 1;

}

**if** (shape == **'2'**) {

realShape = 2;

}

**if** (shape == **'3'**) {

realShape = 3;

}

**if** (shape == **'4'**) {

realShape = 4;

}

cout << **"Enter the dimension: "**;

cin >> dim;

cout << **"Note: Only these characters will be assumed as valid characters: "**;

cout << **"\* ? % + $ # ~ @ ( )"** << endl;

cout << endl;

**do** {

cout << **"Enter the character type: "**;

cin >> letter;

} **while** (letter != **'\*'** && letter != **'?'** && letter != **'%'** && letter != **'+'** && letter != **'$'** && letter != **'#'** &&

letter != **'~'** && letter != **'@'** && letter != **'('** &&

letter != **')'** && cout << **"Invalid Input, Please Enter any valid input:"** << endl);

**switch** (input) {

**case** 1:

solShapes(realShape, dim, letter);

**break**;

**case** 2:

holShapes(realShape, dim, letter);

**break**;

**case** 3:

solShapes(realShape, dim, letter);

holShapes(realShape, dim, letter);

**break**;

}

cout << **"Do you want to draw more (Y/N): "**;

cin >> rep;

**if** (rep == **"N" or** rep == **"n" or** rep == **"no" or** rep == **"NO" or** rep == **"No" or** rep == **"nO"**) {

**return** 0;

} **else goto** again;

}

#### Code (Function.h):

(Note: Function prototypes are written in function.h . This line is not part of the program.)

#ifndef \_FUNCTIONS\_H

#define \_FUNCTIONS\_H

**void** solRightTri(**int** dim, **char** letter);

**void** solShapes(**int** shape, **int** dim, **char** letter);

**void** solIsoTri(**int** dim, **char** letter);

**void** solEqTri(**int** dim, **char** letter);

**void** holRightTri(**int** dim, **char** letter);

**void** holShapes(**int** shape, **int** dim, **char** letter);

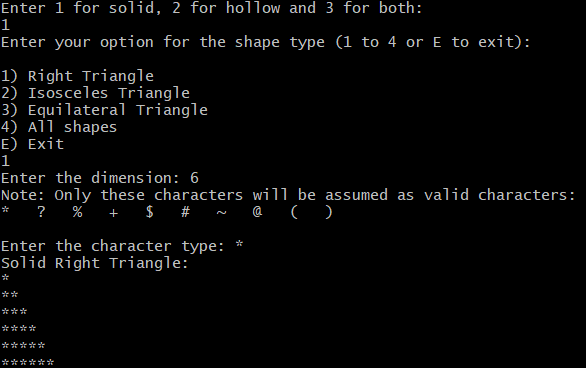
**void** holEqTri(**int** dim, **char** letter);

**void** holIsoTri(**int** dim, **char** letter);

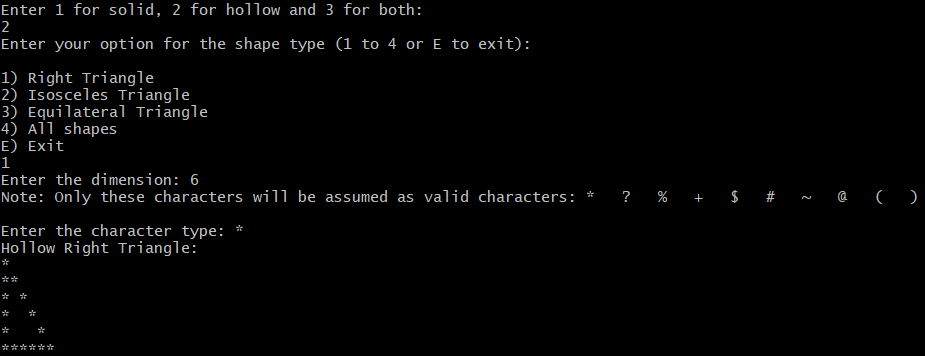
#endif

### Outputs:

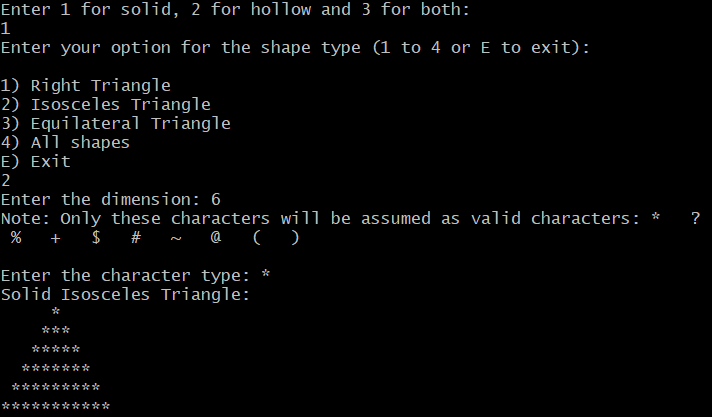
##### Solid Right Triangle:



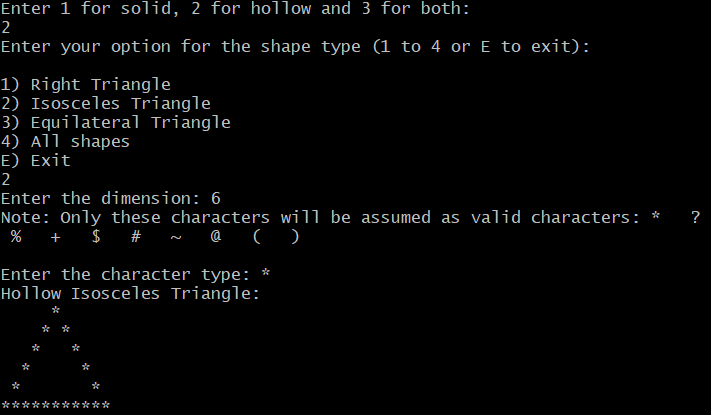
##### Hollow Right Triangle:



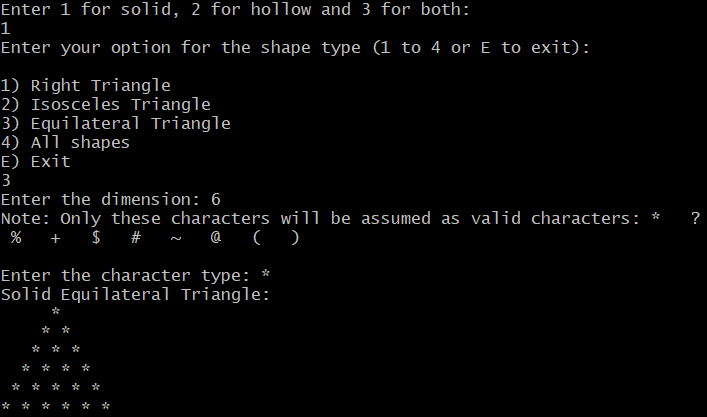
##### Solid Isosceles Triangle:



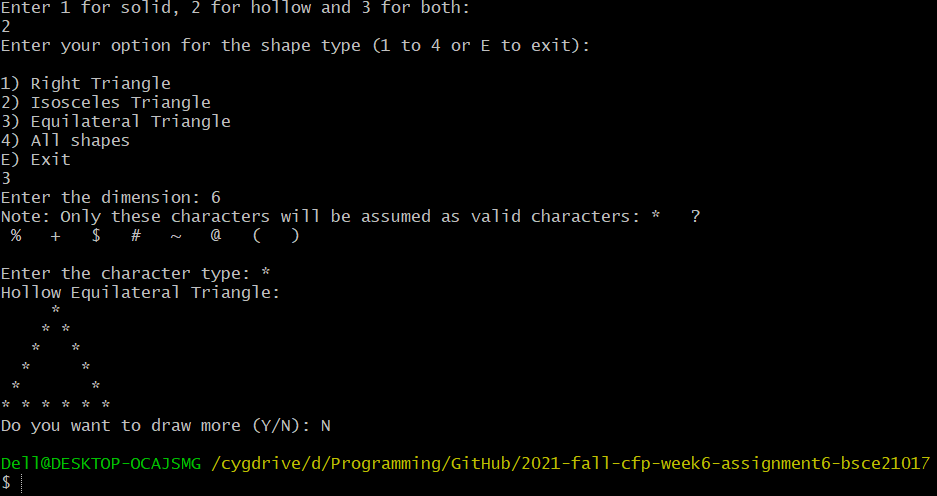
##### Hollow Isosceles Triangle:



##### Solid Equilateral Triangle:



##### Hollow Equilateral Triangle:



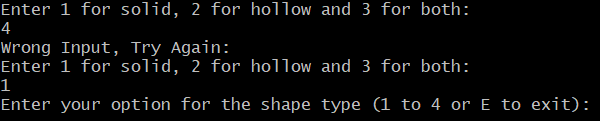
###### Note:

*Program is ended as the user commanded the program to end.*

##### Errors Possibilities by Users:

###### Possibility 1:

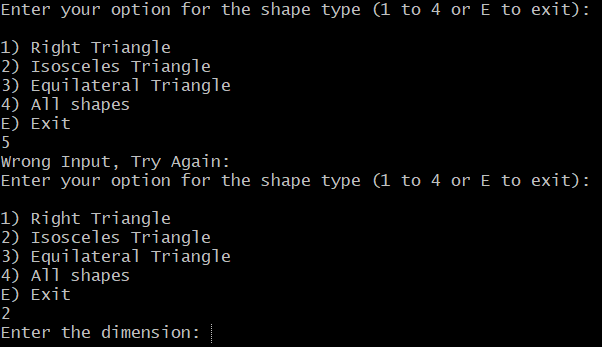
Program will continue asking for the valid input if the input is other than 1, 2 or 3



Note that first, the user gave 4 as input which is irrelevant for that. So the program told the user to enter the value again for Input . Then the user entered 1 which is one of the correct options then the program proceeded forward.

###### Possibility 2:

In the following case, Program will continue asking for the valid input if input is other than 1, 2, 3, 4 or E.

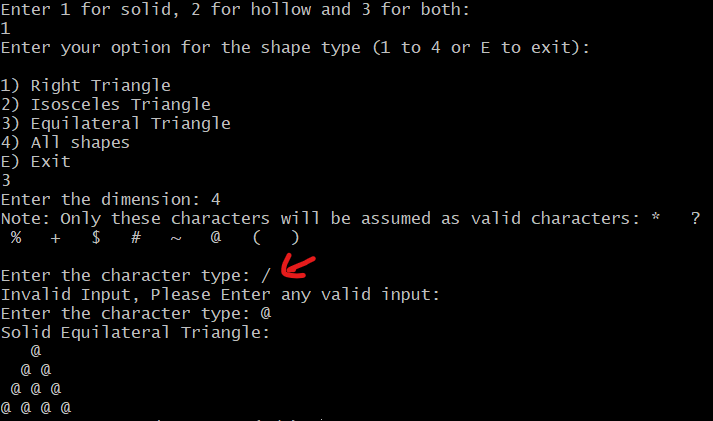


Note that first, the user gave 5 as input which is not the valid input for that. So the program notified the user to enter the valid input. Then the user inputed 1 which is one of the correct options, then the program proceeded forward.

###### Possibility 3:

For following part of the program, only these characters will considered as valid characters by the program: **\* ? % + $ # ~ @ ( )**

*program will continue asking for the valid input if input is other than these characters.*

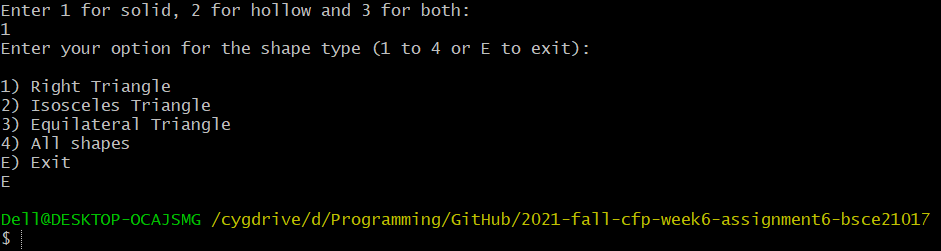
**

Note that first, the user gave “/” as input which is not valid in this case. So the program warned the user to enter the correct one. Then the user entered “@” which is valid, that’s why the program moved forward.

##### Other Possibilities for User:

###### ‘Exit’ Possibility:

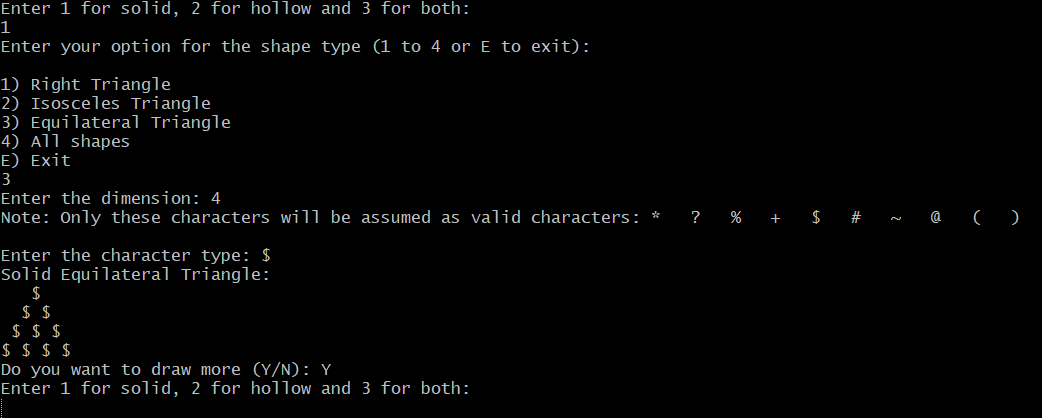
If the user selects “E” in this case, the program will end without displaying the Output.



User selected the “E” option and as a result the program ended without any further advancement.

###### ‘Restart’ Possibility:

If the user wants, after displaying the output, to again start the program from the beginning. He can do it. But if the user wants to end the program at that point, he has the privilege for that.



After completing the program once, the program asked the user whether he wants to draw again or wants to end. User entered “Y” (Yes) then the program started again from the beginning.

*Note: In screenshot attached with hollow equilateral triangle, User told the program to end. As a result, the program ended.*