Tshwane University of Technology
We empower people

FORMAL SUMMATIVE ASSESSMENT INSTRUCTIONS:

1. All rules and regulations regarding student conduct and behaviour, as agreed upon by students at registration and as otherwise stated by the Tshwane University of Technology, apply.

2. A student card or proof of registration, along with an original identification document, must be presented to invigilators.

3. Ensure that your student number is correctly indicated on all submissions (paper-based or online) and that the attendance record is signed as required.

4. If a student is suspected of any form of cheating or plagiarism, either during the assessment or afterwards, the examiner, in their professional judgment and after consultation with the appointed moderator, may, at minimum, award a zero mark for the assessment. Further action may be taken against the student.

DEPARTMENT OF
ELECTRICAL ENGINEERING

FACULTY OF
ENGINEERING AND THE BUILT ENVIRONMENT

MODULE CODE
ES216AB

MODULE NAME
ENGINEERING SOFTWARE DESIGN

ASSESSMENT NAME
Evaluation 3 B

EXAMINER:	Mr D Engelbrecht Prof J.A. Jordaan	**MARKS:**	65 points
MODERATOR:	Mr A.J. Smith	**PAGES:**	15 (incl. cover)
DATE:	MAY 2025	**TIME:**	2 hours

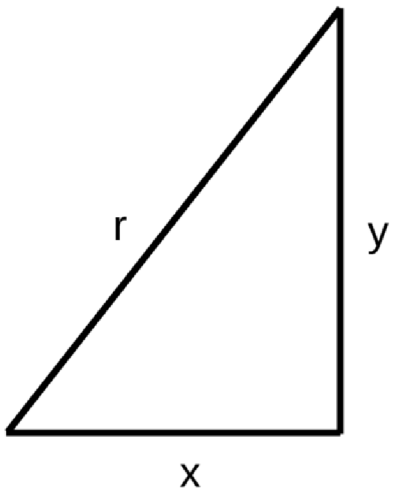
STUDENT NUMBER:								
SURNAME:			**INITIALS:**			**SIGNATURE:**		

NOTE THE FOLLOWING:

1. **Plagiarism Policy:** Original work is required. We will use similarity detection software to review all student submissions for plagiarism. Ensure your work is your own.
2. **Internet Protocol (IP) Tracking:** IP addresses will be recorded and checked to verify that you have uploaded your work from the correct TUT laboratory.
3. **No External Devices:** The use of USB or other external devices is prohibited during the evaluation.
4. **Internet Access:** External internet access is not permitted.
5. **Evaluation Content:** This evaluation will cover topics from Unit 1 to Unit 6
6. **Programming Language:** Write your program in C, adhering to structured programming principles.
7. **Editing Requirements**: Your program must comply with all specified requirements. Refer to the appendices and attachments for more details.
8. **Submission Format:** Submit your source code file in the format
   <student number>.c , for example, 217123456.c
   (ONLY YOUR STUDENT NUMBER! Do not add your other text.)
9. **Submission Upload:** Use the dedicated upload link on MyTUTorD2L to upload your C code only. While multiple submissions are allowed, only the latest submission will be retained. If you upload the wrong file by mistake, simply re-upload the correct one, and the previous submission will be overwritten.
10. **Backup And Save:** Remember to save your work on the PC D:Drive and save regularly throughout the evaluation. In the event of PC malfunction or power failure, only 5 to 10 minutes (depending on the case) extra time will be allotted.

QUESTION:

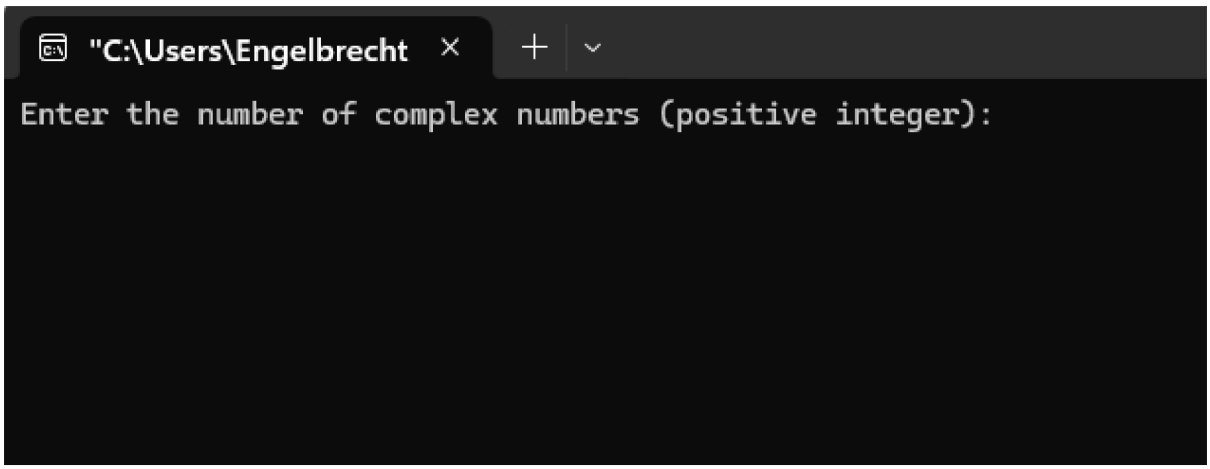
Question Create a structured C program for a Right Angle Triangles Management Menu. See the figure below for the notation used.



Requirements
Hint: Use the sqrt() function to calculate r.

Before proceeding to the menu-driven section of the application, perform the following steps:

1. Prompt the user to enter a size for the 3 dynamic arrays (x, y, r)
2. Create three dynamic arrays (`x`, 'y' and `r`) of the same size using `malloc()` or `calloc()`, based on the size provided by the user.



Within the main function (after the dynamic memory allocation), a **switch statement** should be used to handle user selections. This switch statement should be nested within a loop, allowing the user to make multiple selections until they choose to exit.

Application functions to be implemented and corresponding user selections:
Function prototypes:

```
int displayMenu(void);
void generateRandomNumbers(int* x, int* y, int* r, int size);
void displayArrays(int* x, int* y, int* r, int size);
void sortDescending(int* x, int* y, int* r, int size);
void sortAscending(int* x, int* y, int* r, int size);
void findMaxMin(int* r, int size);
void linearSearchY(int* x, int* y, int* r, int size, int value);
```

Function 1: DisplayMenu

Description:
This function displays the main menu options to the user and prompts them to make a choice. It then returns the user's choice as an integer..

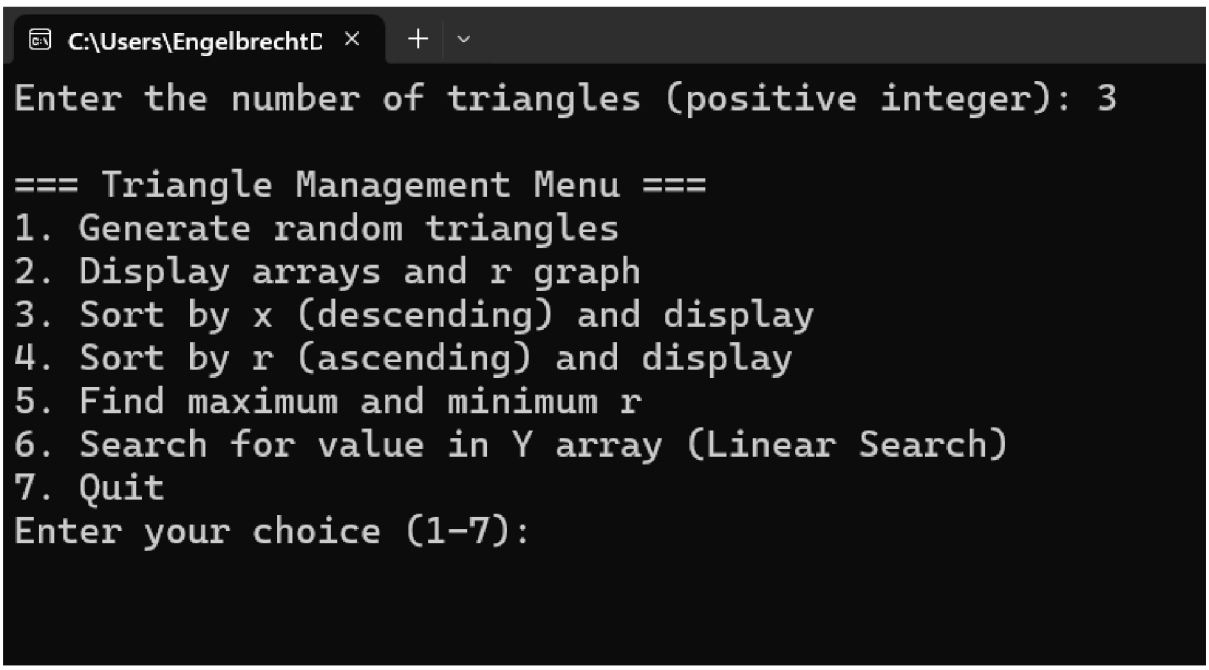
Details:

- The function does not take any parameters.
- It prints a list of menu options (numbered 1 - 7) using `printf`.
- It uses `scanf` to read the user's choice and stores it in a local variable `choice`.
- The function returns the value of `choice`, which corresponds to the menu option selected by the user.

Function Prototype:
int DisplayHeading (void);

Example Call:
int choice = DisplayMenu();

Output:



Function 2: Generate Random Numbers

Description:

This function generates random numbers for the x and y arrays(1 - 50). Populate the ‘r’ array with the formula sqrt(x^2+y^2)

Details:

- Parameters:
  - `x` (int*): An array of integers representing the real numbers
  - `y` (int*): An array of integers representing the imaginary numbers
  - `r` (int*): An array of integers representing the magnitude numbers
  - `size` (int): The number of complex numbers.
- The function iterates through each array index and generates a random number for the real numbers(1 - 50) and imaginary numbers (1 - 50)
- The function calculates the magnitude of the real and imaginary to populate the magnitude array.

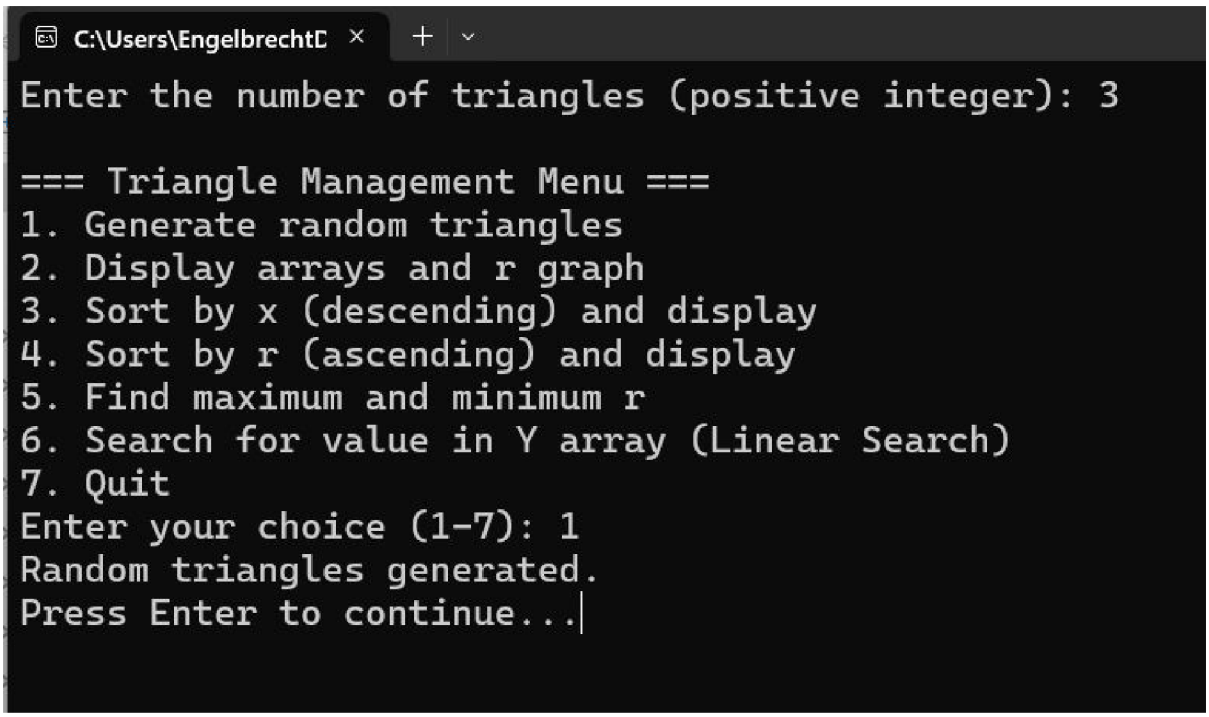
Function Prototype:

void generateRandomNumbers(int* x int* y, int*r, int size)

Example Call:

generateRandomNumbers(x, y, r, size)

Output:



Function 3: Display Arrays and Graph

Description:

This function displays a graphical representation of the ‘r’ array. It prints all the values of the x, y and r array followed by the graph.

Details:

- Parameters:
  - `x` (int*): An array of integers representing the real numbers
  - `y` (int*): An array of integers representing the imaginary numbers
  - `r` (int*): An array of integers representing the magnitude numbers
  - `size` (int): The number of complex numbers.
- The function prints the values of each array.
- The function prints * equal to ‘r’.

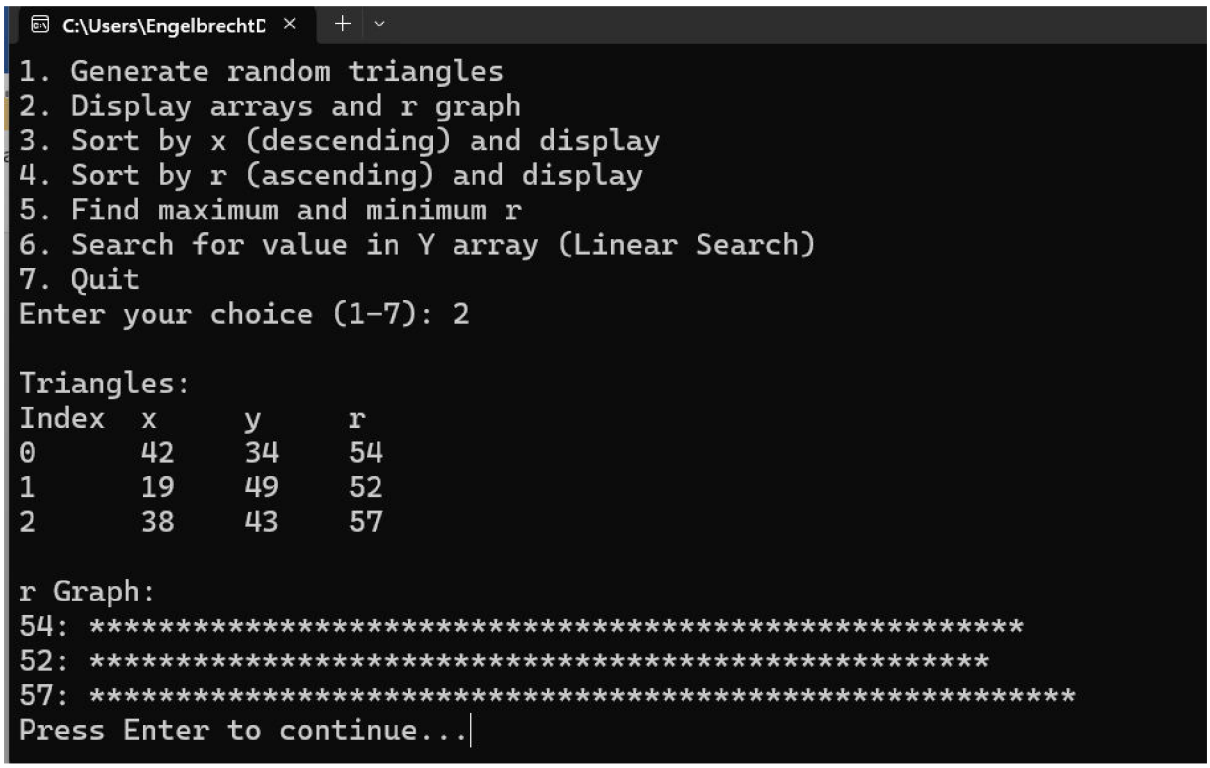
Function Prototype:

void displayArrays(int* x, int* y, int* r, int size)

Example Call:

displayArrays(x, y, r, size)

Output:



Function 4: SortDescending

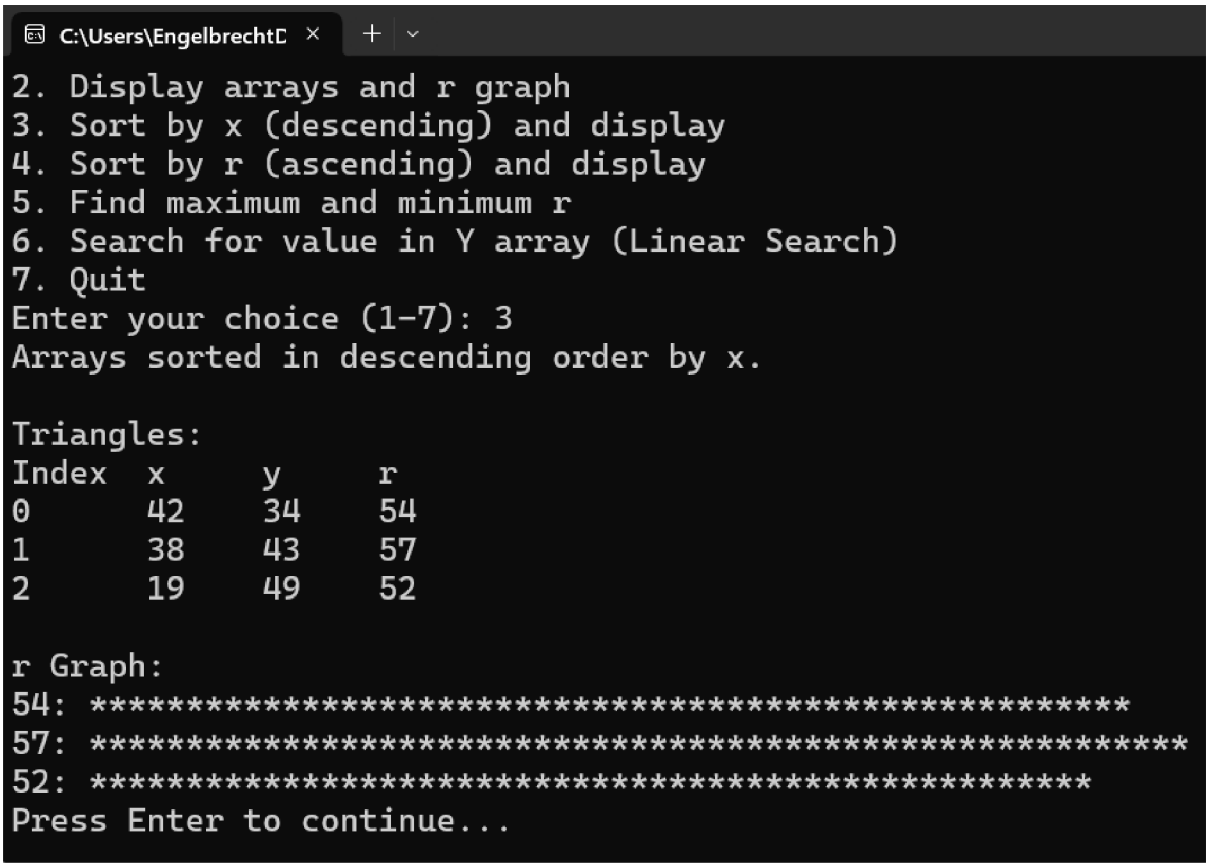
Description:
This function sorts the ‘x’ array in descending order, highest to lowest and displays the ‘r’ graph

Details:

- Parameters:
  - `x` (int*): An array of integers representing the real numbers
  - `y` (int*): An array of integers representing the imaginary numbers
  - ‘r` (int*): An array of integers representing the magnitude numbers
  - `size` (int): The number of complex numbers.

- The function uses nested loops to repeatedly compare and swap elements to sort the arrays.

Function Prototype:
void sortDescending(int* x, int* y, int* r, int size)
Example Call:
sortDescending(x, y, r, size)
Output:



Function 5: SortAscending

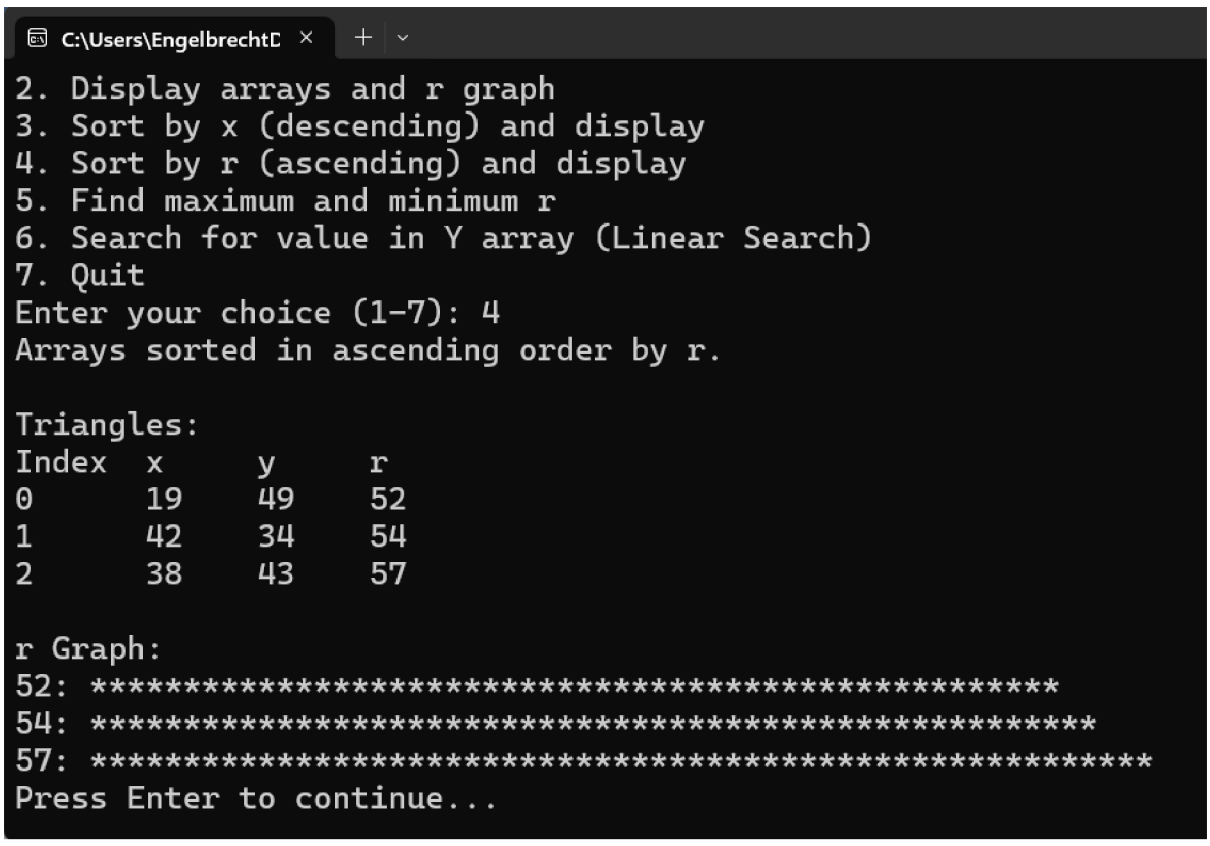
Description:
This function sorts the ‘r’ array in ascending order, lowest to highest

Details:

- Parameters:
  - `x` (int*): An array of integers representing the real numbers
  - `y` (int*): An array of integers representing the imaginary numbers
  - `r` (int*): An array of integers representing the magnitude numbers
  - `size` (int): The number of complex numbers.

- The function uses nested loops to repeatedly compare and swap elements to sort the arrays.

Function Prototype:
void sortAscending(int* x, int* y, int* r, int size)
Example Call:
sortAscending(x, y, r, size)



Function 6: Search Max and Min

Description:
This function searches for the ‘r’ array with the highest and lowest values and prints them.

Details:

- Parameters:
  - `r` (int*): An array of integers representing the values of the magnitudes
  - `size` (int): The number of soft drinks.
- The function initializes `low` to the first element of `r` and `index` to 0.
- The function initializes `high` to the first element of `r` and `index` to 0.
- It iterates through the array to find the smallest value and updates `low` accordingly.
- It iterates through the array to find the largest value and updates `low` accordingly.
- The function prints the smallest and largest values of the ‘r’ array.

Function Prototype:
void findMaxMin(int* r, int size)

Example Call:
void findMaxMin(r, size)

```
=== Triangle Management Menu ===
1. Generate random triangles
2. Display arrays and r graph
3. Sort by x (descending) and display
4. Sort by r (ascending) and display
5. Find maximum and minimum r
6. Search for value in Y array (Linear Search)
7. Quit
Enter your choice (1-7): 5
Maximum magnitude: 57
Minimum magnitude: 52
Press Enter to continue...
```

Function 6: LinearSearch

Description:

Use the Linear Search algorithm to find all positions where the value occurs in the ‘y’ array.

Details:

- Parameters:
  - `y` (int*): An array of integers representing the real numbers
  - `size` (int): The number of complex numbers.
  - ‘value’ (int): The value that the user want to search for.
- The function iterates through the `y` array to find a match for `value`.
- If a match is found, it prints the index and the value found.
- If no match is found, it prints “No value found”

Function Prototype:

void linearSearchImag(int* y, int size, int value)

Example Call:

void linearSearchImag(y, size, value)

```
=== Triangle Management Menu ===
1. Generate random triangles
2. Display arrays and r graph
3. Sort by x (descending) and display
4. Sort by r (ascending) and display
5. Find maximum and minimum r
6. Search for value in Y array (Linear Search)
7. Quit
Enter your choice (1-7): 6
Enter value to search in y array (1-50): 43
Positions of 43 in y array:
Index 1: x=38  y=43  r=57)
Press Enter to continue...
```

Ensure all functions mentioned above are self-written and listed in the specified order below the main function, following the function prototypes. Avoid the use of global variables, apply system("cls") and getch() for user-friendliness, and maintain clear indentation and comments for readability.

Implement The Following Flow Format:

1. Libraries
2. Function prototypes (1 to 7) - Given
3. Main function
    A. Variables
    B. Create Dynamic List
    C. Main Application Menu
        i. Do While
            1. Switch Statement
    D. Free Dynamic Memory
    E. Return 0
4. Function Implementation
    1. DisplayMenu Function
    2. GenerateRandomNumbers Function
    3. DisplayArraysandGraph Function
    4. SortDescending Function
    5. SortAscending Function
    6. MinMax Function
    7. LinearSearch Function

Adherence to the provided instructions, function naming conventions, and structured programming principles is crucial for full marks, even if the program functions correctly. Proper indentation and comments are also essential for clarity and understanding.

```
int displayMenu(void);
void generateRandomNumbers(int* x, int* y, int* r, int size);
void displayArrays(int* x, int* y, int* r, int size);
void sortDescending(int* x, int* y, int* r, int size);
void sortAscending(int* x, int* y, int* r, int size);
void findMaxMin(int* r, int size);
void linearSearchY(int* x, int* y, int* r, int size, int value);
```

ANNEXURE A – MARK ALLOCATION

Note: Score range is 0 - 4 which is: 0-none, 1-poor, 2-average, 3-good, 4-excellent

TEST RUBRIC	SCORE [0-4]	WEIGHT [%]
C CODE EVALUATION - Basic Logic		**65**
1. Overall Neatness, Indentation, and Spacing		3
2. User Defined Dynamic Arrays		6
3. Menu Driven: Do While and Switch		5
4. User Friendly Menu: Screen Clear And Input Control		3
5. Free Dynamic Memory		3
6. Function1: DisplayMenu		4
7. Function2: Generate Random Numbers		6
8. Function3: DisplayGraph		8
9. Function4: SortDescending		6
10. Function5: SortAscending		6
12. Function6: MinMax Function		6
12. Function7: Linear Search		4
14. No Runtime or Compile Errors		5
TOTAL		**65**

Graduate Attribute	GA Number	GA Score [0-5]
Engineering Professionalism	GA10	
Application of scientific and engineering knowledge	GA2	
Engineering methods, skills, tools, including information technology	GA5	
Impact of Engineering Activity	GA7	

ANNEXURE B – INFORMATION SHEET

Libraries:	<stdio.h> , <stdlib.h> , <time.h> , <math.h>
Data types:	void, char, short, int, float, double
Data Type modifiers:	const, auto, static, unsigned, signed
Arithmetic operators:	* / % + -
Relational operators:	< <= > >= == !=
Assignment operator:	= += -= *= /= %= &= ^= |= <<= >>=
Logic operators:	&& || !
Bitwise logic operators:	& | ^ ~ << >>
Pointer operators:	Derefernce: * Address: &
Control Structures:	
IF Selection:	if (condition) { … };
IF ELSE Selection:	if (condition) { … } else { … };
SWITCH Selection:	switch (control variable) { case 'value': … ; break; default: … ; break; }
FOR Loop:	for (initial value of control variable; loop condition; increment of control variable) { … }
WHILE Loop:	while (condition) { … };
DO WHILE loop:	do { … } while (condition);
Functions:	return_data_type function_name (parameters) { … };
Common Library Functions:	printf() , scanf() , rand() , srand() , time() , isalpha() , isdigit() , getchar() , getch(), strcpy()
Arrays:	One dimensional:data_type variable_name[size]; Two dimensional: data_type variable_name [x_size][y_size];

ANNEXURE C – ASCII TABLE

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL