Tshwane University of Technology

We empower people

INSTRUCTIONS TO CANDIDATES

1. All exam rules stated by the Tshwane University of Technology apply.
2. **Ensure a single final version of your source code is handed in as requested.**
3. If needed, state all necessary assumptions clearly in code commentary.

MARKS: 100%

PAGES: 15 (incl. cover)

EXAMINER:

Mr A.J. Smith

Prof J.A. Jordaan

MODERATOR:

Mr D Engelbrecht

TIME:

90 Minutes

(30 minutes extra time)

FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT

DEPARTMENT OF ELECTRICAL ENGINEERING

ES216BB ENGINEERING SOFTWARE DESIGN B

EVALUATION 1

SEPTEMBER 2024

EVALUATION INSTRUCTIONS

1. **Plagiarism:** Submit only original work. We will use similarity software to verify the authenticity of all submissions.

2. **Permitted Tools:** You are allowed to use only **CodeBlocks,** and **Google Chrome** to access the evaluation, view the evaluation PDF and upload submission for this evaluation. Access to emails, other online resources, and memory sticks is strictly prohibited. Please be aware that computer activity will be remotely monitored. Breaches of TUT's official examination and module rules will result in a minimum penalty of zero for this evaluation, with the potential for further disciplinary action.

3. **File Submission:** Your source code file must be named according to this format: **"<student number>.h" (e.g. 21011022.h).** Do not add any other text (name, surname, etc.) to the file name (ONLY YOUR STUDENT NUMBER).

4. **Uploading Instructions:** Submit your ".cpp" file via the designated upload link. While multiple uploads are allowed, only the most recent submission will be retained on the system. If you make an error in your initial upload, simply re-upload your file, and the previous version will be overridden. Download your submission to confirm that the correct file has been uploaded.

5. **Evaluation Scope:** This assessment encompasses basic content from ES216AB and specifically ES216BB content defined in **Unit1 to Unit3**

6. **Programming Language:** Construct your program in **C++** and adhere to structured programming principles.

7. **Editing and Requirements:** Your program must meet all specified requirements. Refer to the attached appendices for additional details.

8. **Evaluation Requirements:**

   a. Remember to save your work on the PC "D: Drive" and save regularly throughout the evaluation.
   b. Do not modify the given code in the ".cpp" file except for implementing the requested functions as required.
   c. Use the exact function names and parameters as used in the in the ".cpp" file function prototypes and main function.
   d. Complete the C++ functions below the main function in each comment block as shown.

C++ FILE CODE EXPLANATION

The provided C++ code is designed to manage a dynamically created array of structures, where each structure contains a string and a float value. The code starts by including necessary standard header files such as “**iostream**” and “**fstream**”. The “**using namespace std**” directive is used to simplify the code by avoiding the need to prefix standard library entities with “**std::**”.

A “**struct**” named “**Record**” is defined to represent each element of the dynamically allocated array. Each `Record` contains a string (“**sData**”) and a float (“**fData**”). The functions that manage this data will be implemented below the main function.

The “**main**” function initialises a pointer to the dynamically allocated array, “**records**”, and sets it to “**nullptr**” or “**NULL**” (depending on the compiler settings), indicating an empty array. Variables such as “**arrSize**”, “**choice**”, “**FileName**”, and “**Average**” are declared to store the array size, the user's menu choice, the name of the file to be read, and the average of the float values in the array, respectively.

The program then enters a do-while loop, which displays a menu to the user with options to read and populate the array from a file, calculate the average of the float values, display the string data, display the float data along with its deviation from the average, and delete the array and exit the program. The user is prompted to enter a choice, and based on the input, a switch-case construct is used to call the appropriate function.

For instance, if the user selects the first option, they are prompted to enter a filename, and the “**ReadTextFile**” function is called to populate the array. If the second option is chosen, the “**AverageArray**” function is called to compute the average of the float values, and the result is displayed. The third and fourth options call the two versions (overloaded) of the “**DisplayRecordData**” function to show the data in the array. The fifth option deletes the array using the “**DeleteArray**” function and exits the program. If the user enters an invalid choice, a message is displayed.

After each operation, the user is prompted to press any key to continue, and the screen is cleared using the “**system("cls")**” command. The loop continues until the user selects the fifth option to exit.

HEADER FILE FUNCTIONS

The required functions for the C++ program should be implemented in the appropriate comment blocks as given in the .cpp file. The function declarations and descriptions are as follows:

1. Initialise Record Member Function

void Record::InitialiseRecord(string sPara, float fPara);

The **InitialiseRecord** function is a member of the **Record** structure. It is used to initialise the **sData** and **fData** members of the Record structure with the provided parameters. The function receives two parameters: a string **sPara** and a float **fPara**, which are used to set the **sData** and **fData** members of the Record, respectively. This function populates a r**ecord** structure element with data.

2. Text File Line Count Function

int TextFileLineCount(string FileName);

The **TextFileLineCount** function is designed to count the number of lines in the text file, which represent one data point entry. It has a file name parameter, string **FileName**, that contains the text file name and the file type extension. The function opens the text file and then traverses through the file by counting each line using the **getline()** function, which receives the file object and string variable as parameters. Thereafter, the file must be closed appropriately before the line count is returned to the function call.

3. Read File and Populate Function

void ReadFileAndPopulate(string FileName, Record **records, int *arrSize);

The **ReadFileAndPopulate** function reads data from a text file and populates a dynamically allocated array of Record structures. It receives three parameters: the file name, string **FileName**, a double pointer, Record ****records**, to the records array, and a pointer, int **arrSize**, that stores the size of the array.

1. The function first checks if any memory is allocated for the records array and deallocates it, if necessary, before continuing.
2. Counts the number of lines in the file using the **TextFileLineCount** function.
3. Allocates memory for the records array based on the count.
4. If memory allocation fails, an error message is displayed.
5. Reads the data from the file to populate each Record structure in the array using the structure member function **InitialiseRecord**.

4. Average Array Function

float AverageArray(Record *records, int arrSize);

The **AverageArray** function calculates the average of the **fData** values in an array of Record structures. It receives two parameters: a pointer to the records array and an integer representing the size of the array. The function sums up all the **fData** values in the array and divides the total by the number of elements in the array to calculate the average. **If the array size is zero**, the function returns **0.0** to avoid division by zero.

5. Display Record Data Function 1 - Overloaded

void DisplayRecordData(Record *records, int arrSize);

The **DisplayRecordData** function is used to display the contents of an array of Record structures in a formatted manner. It receives two parameters: a pointer to the records array and an integer representing the size of the array. The function outputs the sData and fData of each Record in the array in a table format, with appropriate column headings and separators. This function provides a basic display of the data without any additional calculations. The **iomanip** library function **setw()**, is used to effectively style the table output.

6. Display Record Data Function 2 - Overloaded

void DisplayRecordData(Record *records, int arrSize, float avg);

This version of the **DisplayRecordData** function is similar to function 4 above, but includes an additional parameter avg, representing the average of the fData values in the array. The function displays each Record's **sData**, **fData**, and the **deviation** of **fData** from the average. The deviation is calculated as “**fData - avg**“ and is displayed in a third column. The output is formatted in a table with appropriate headings and separators. The **iomanip** library function **setw()**, is used to effectively style the table output.

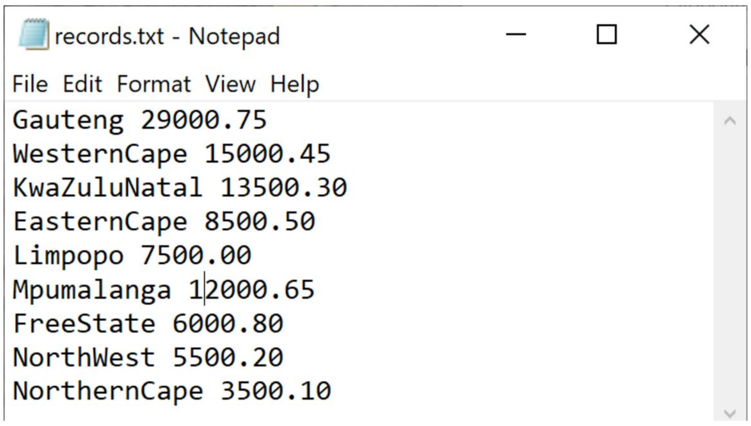
7. Delete Array Function

void DeleteArray(Record **records, int *arrSize);

The **DeleteArray** function is responsible for deallocating the memory allocated for the records array and resetting its **size** to **zero**. It receives two parameters: a double pointer to the records array and a pointer to the size of the dynamic array. The function deletes the dynamically allocated array of Record structures, sets the records pointer to nullptr or NULL to avoid dangling pointers, and resets array size to zero, indicating that the array no longer holds any data.

PRINT SCREENS

Text File Content:

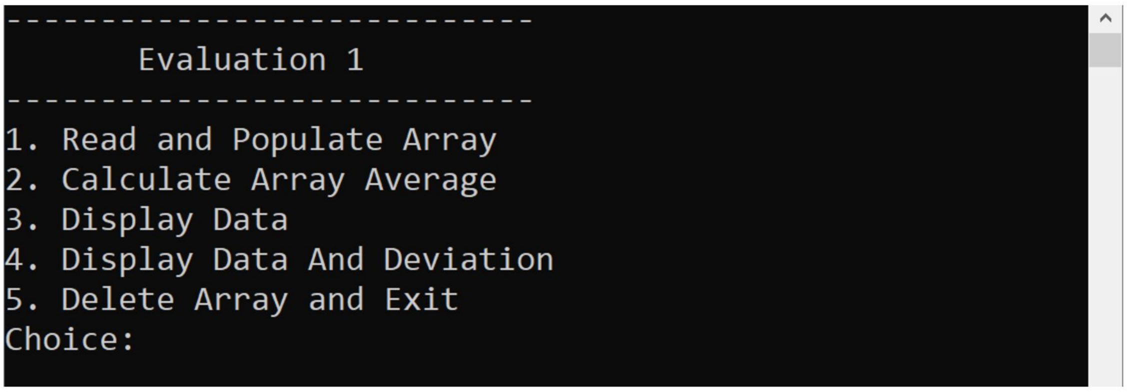


records.txt - Notepad

File Edit Format View Help

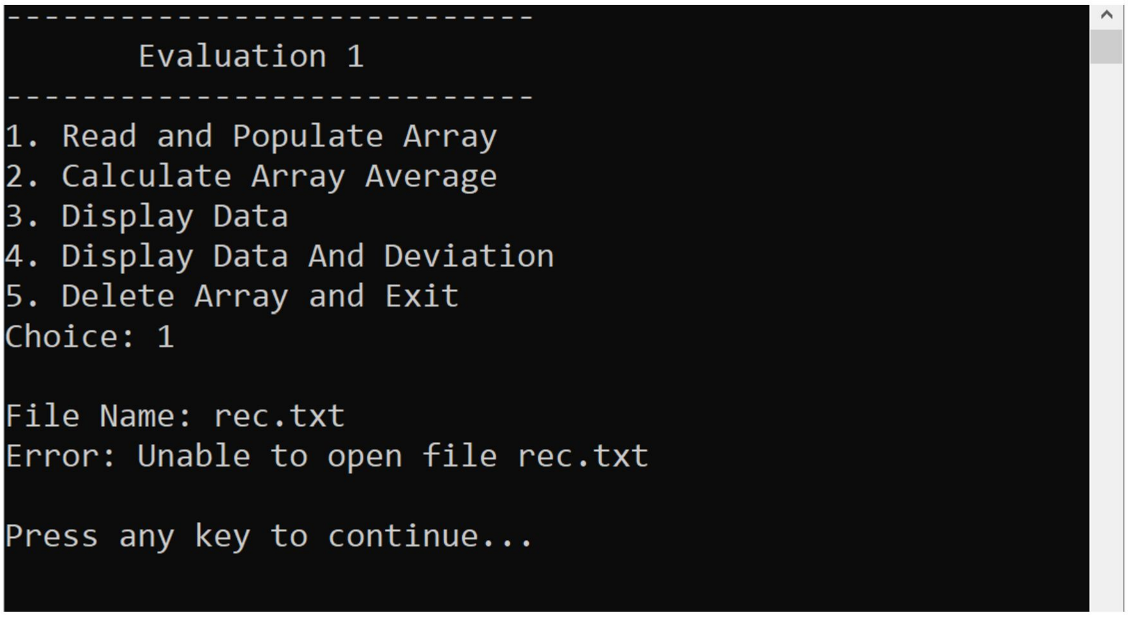
```
Gauteng 29000.75
WesternCape 15000.45
KwaZuluNatal 13500.30
EasternCape 8500.50
Limpopo 7500.00
Mpumalanga 12000.65
FreeState 6000.80
NorthWest 5500.20
NorthernCape 3500.10
```

Main Menu:



```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice:
```

Load Non-Existent File:

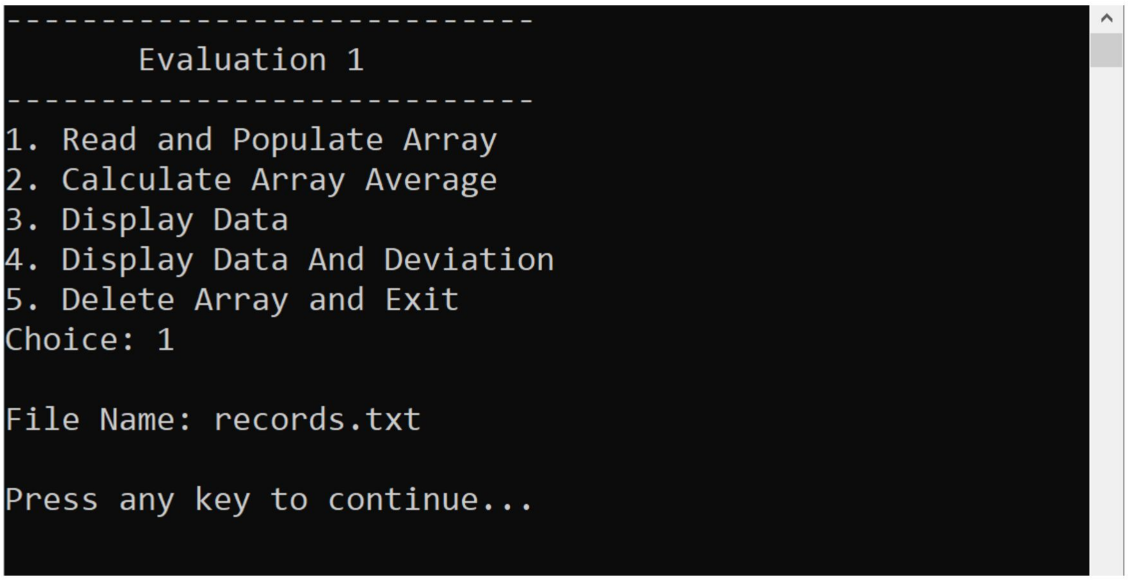


```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice: 1

File Name: rec.txt
Error: Unable to open file rec.txt

Press any key to continue...
```

Load Existent File:

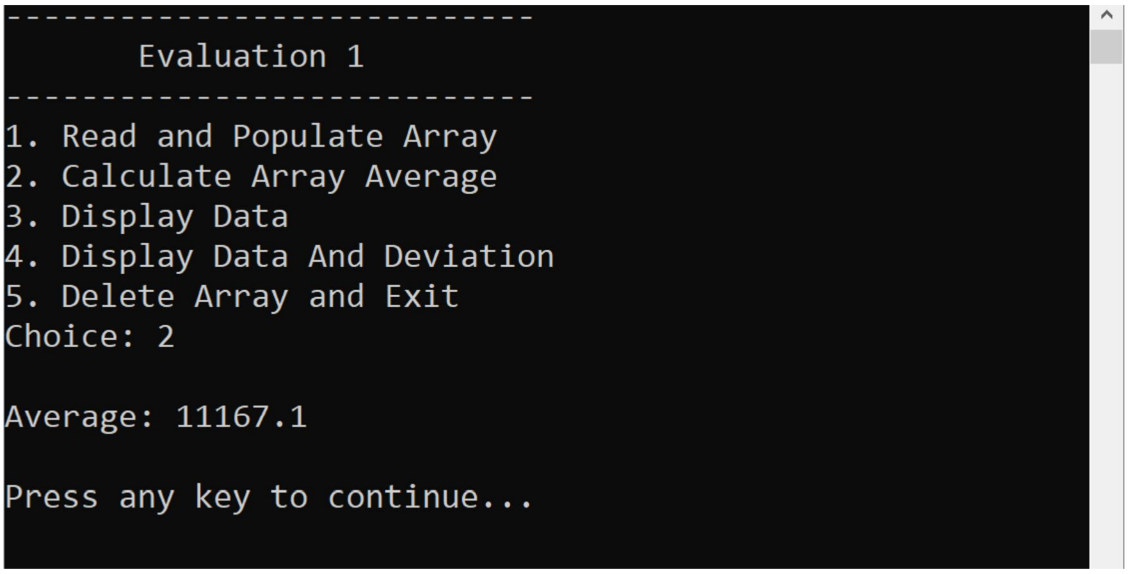


```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice: 1

File Name: records.txt

Press any key to continue...
```

Display Average of Float Data:

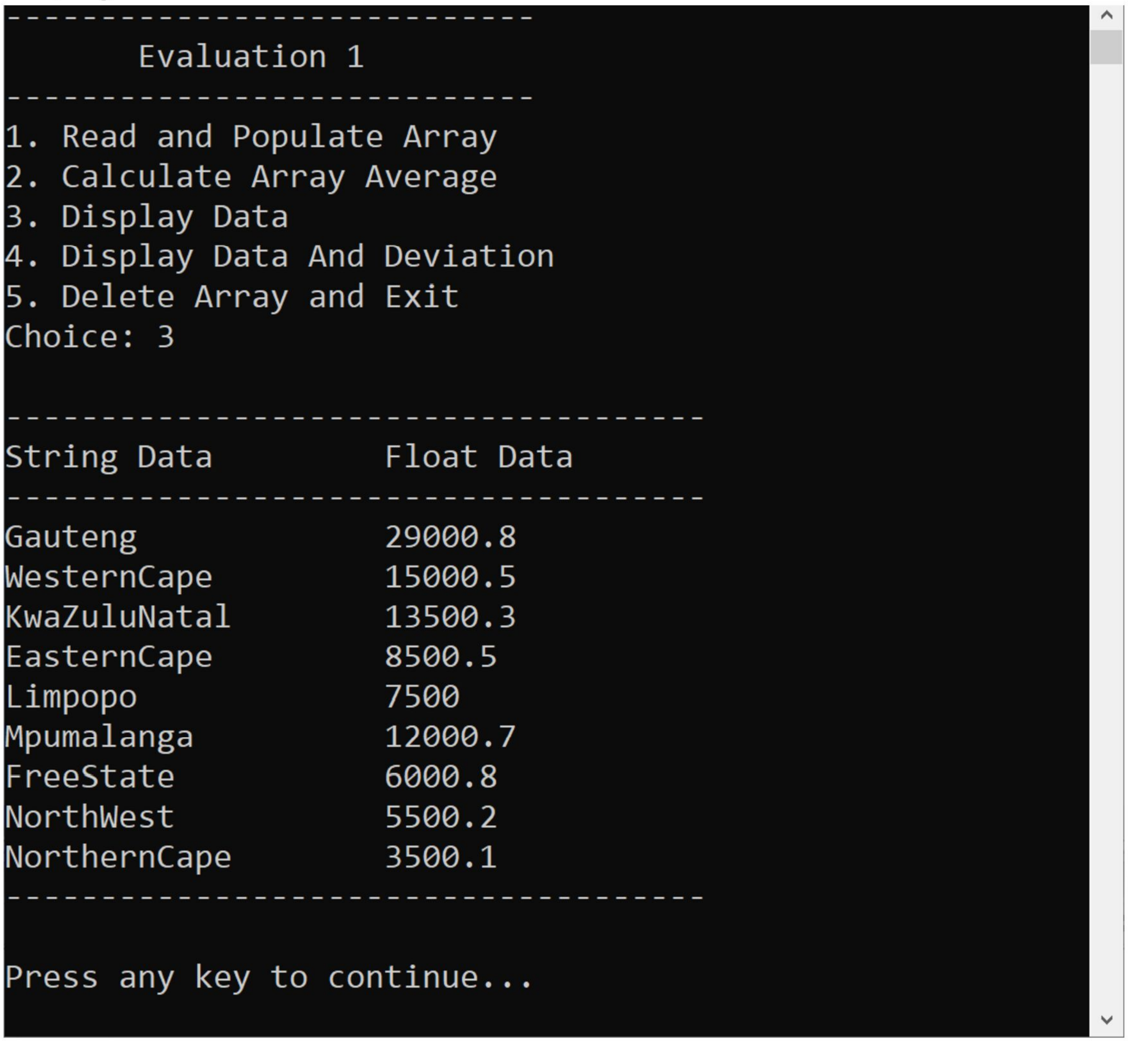


```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice: 2

Average: 11167.1

Press any key to continue...
```

Display records data:

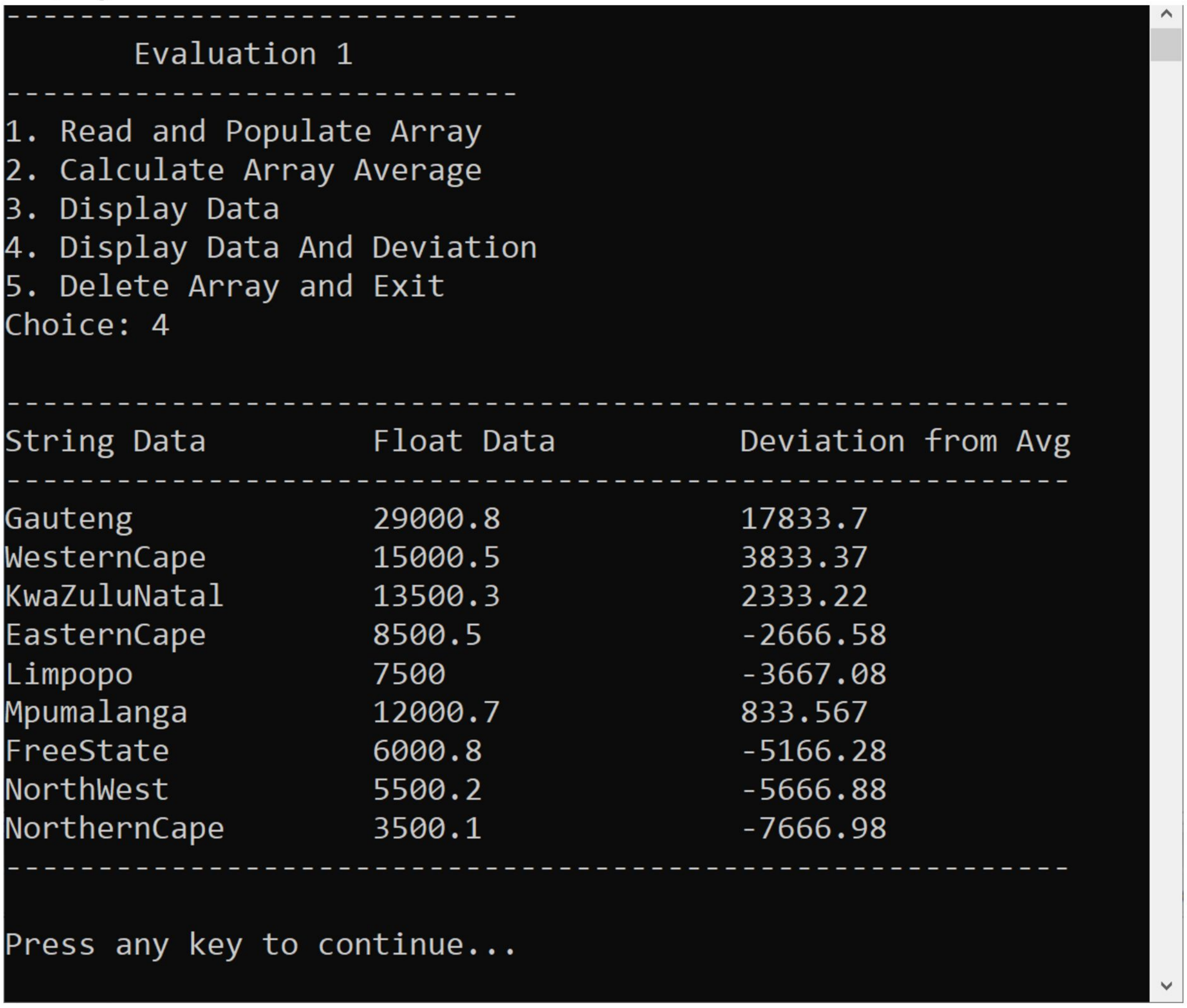


```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice: 3

-------------------------------------
String Data         Float Data
-------------------------------------
Gauteng             29000.8
WesternCape         15000.5
KwaZuluNatal        13500.3
EasternCape         8500.5
Limpopo             7500
Mpumalanga          12000.7
FreeState           6000.8
NorthWest           5500.2
NorthernCape        3500.1
-------------------------------------

Press any key to continue...
```

Display Records Data with Deviation:

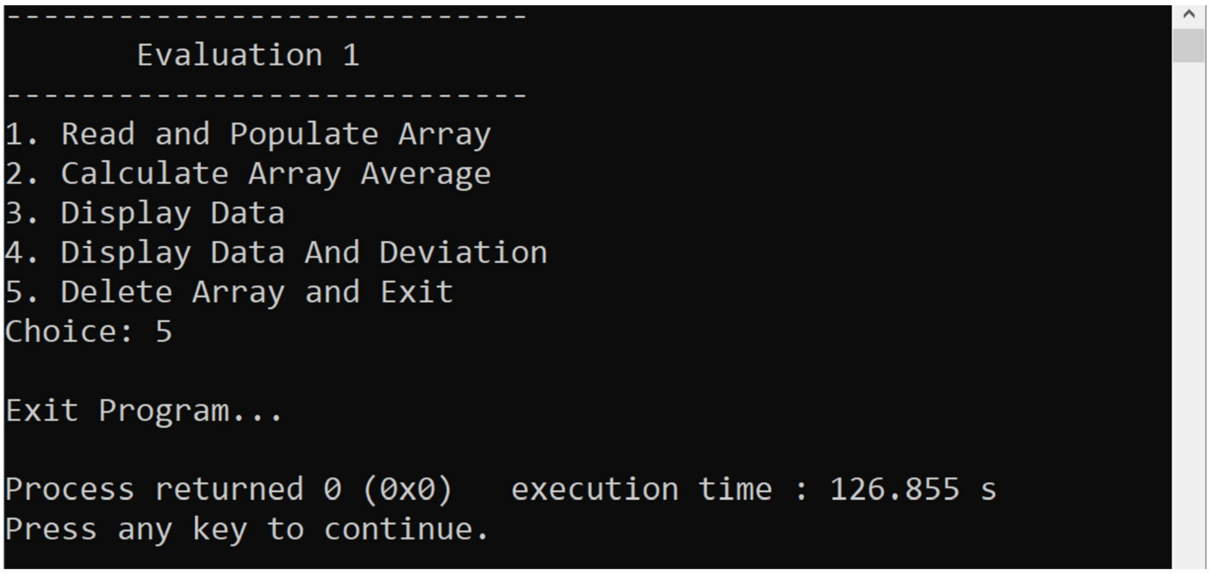


```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice: 4


----------------------------------------------------------
String Data         Float Data          Deviation from Avg
----------------------------------------------------------
Gauteng             29000.8             17833.7
WesternCape         15000.5             3833.37
KwaZuluNatal        13500.3             2333.22
EasternCape         8500.5              -2666.58
Limpopo             7500                -3667.08
Mpumalanga          12000.7             833.567
FreeState           6000.8              -5166.28
NorthWest           5500.2              -5666.88
NorthernCape        3500.1              -7666.98
----------------------------------------------------------


Press any key to continue...
```

Exit:



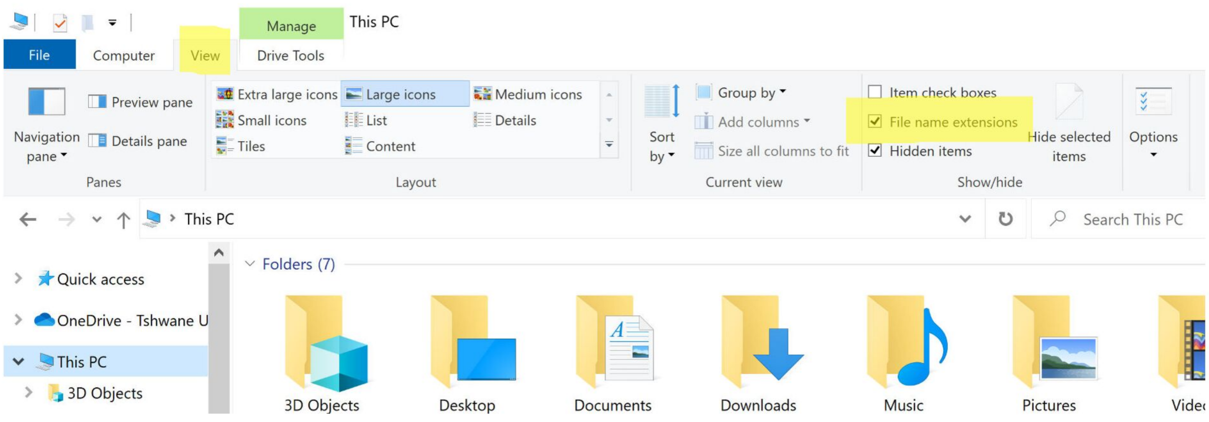
```
----------------------------
       Evaluation 1
----------------------------
1. Read and Populate Array
2. Calculate Array Average
3. Display Data
4. Display Data And Deviation
5. Delete Array and Exit
Choice: 5

Exit Program...

Process returned 0 (0x0)   execution time : 126.855 s
Press any key to continue.
```

HOW TO RUN THE SHOWCASE FILE

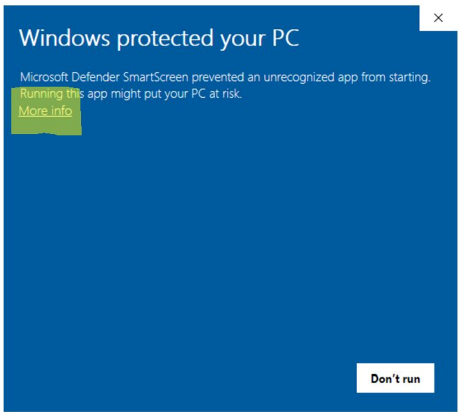
1. Enable file extensions (see highlighted in yellow)



2. Change the name from **“Showcase.old”** to **“Showcase.exe”**

3. Run the **“ShowcaseEV.exe”** by double-clicking on the icon.

4. The following may be shown by Windows. Click on **“More info”**

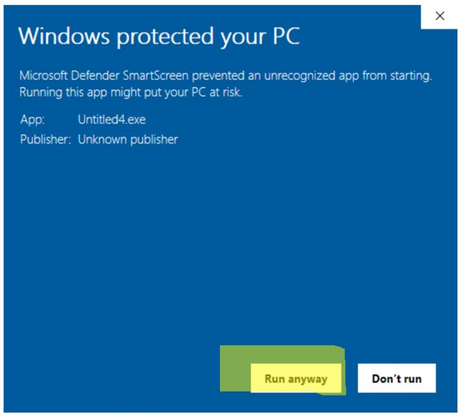


Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

More info

5. Click on **“Run anyway”**



Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: Untitled4.exe

Publisher: Unknown publisher

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		**55**
1. Initialise Record Member Function (Initialize structure members with default parameters)		**5**
2. Count File Lines Function (Insert new record into dynamic array)		**7**
3. Read and Populate Function **3.1 Deallocate Memory Before**		**2**
3. Read and Populate Function **3.2 Retrieve Line Count**		**2**
3. Read and Populate Function **3.3 Allocate Dynamic Memory and set Size**		**3**
3. Read and Populate Function **3.4 Check Memory Allocation Failure**		**2**
3. Read and Populate Function **3.5 Read Text File and Populate Using Initialise**		**3**
4. Average Array Function (Calculate average of float data in array)		**6**
5. Display Record Data Function 1 - Overloaded (Display array data: string and float)		**6**
6. Display Record Data Function 2 - Overloaded (Display array data: string, float, and deviation)		**6**
7. Delete Array Function (Deallocate dynamic array memory)		**5**
9. Overall Impression (Neatness, Readability, Spacing, and Indentation)		**4**
8. No Compile or Runtime errors		**4**
TOTAL		**50+5**

Graduate Attribute	GA Number	GA Score [0-5]
Application of scientific and engineering knowledge	GA2	1, 2, 3
Engineering methods, skills, tools, including information technology	GA5	5,6,7
Impact of Engineering Activity	GA7	9
Engineering Professionalism	GA10	8

ANNEXURE B – INFORMATION SHEET

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 { … };
WHILE Loop:	while (condition) { … };
DO WHILE loop:	do { … } while (condition);
FOR Loop:	for (initial value of control variable; loop condition; increment of control variable) { … }
SWITCH Selection:	switch (control variable){ case ‘value’: … ; break; default: … ; break; }

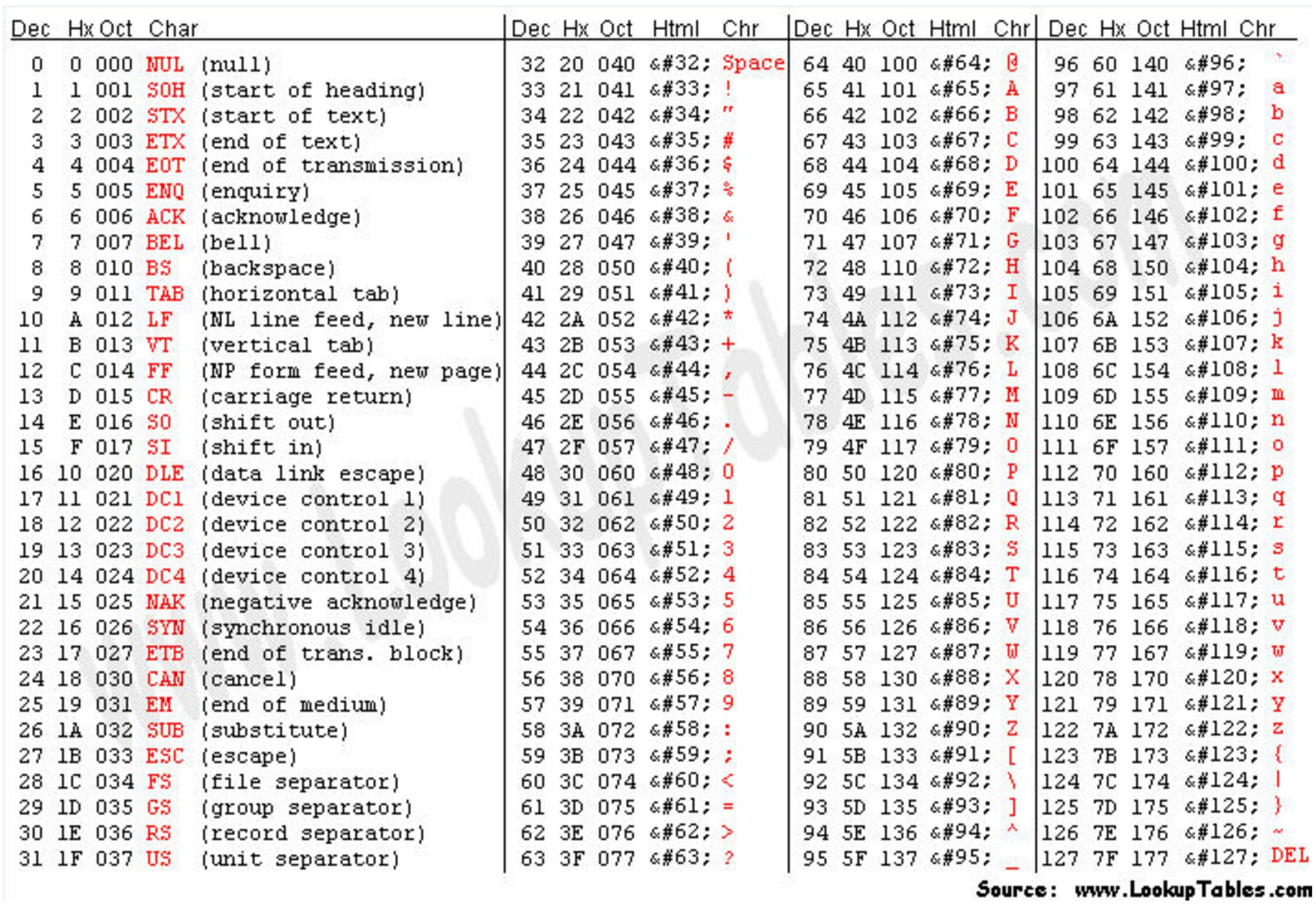
Functions: return_data_type function_name (parameters) { … };

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

Source: www.LookupTables.com