



**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: 13 (incl. cover)

EXAMINER:

Mr A.J. Smith

Prof J.A. Jordaan

MODERATOR:

Mr D Engelbrecht

TIME:

2 hours (30 minutes extra)

ELECTRICAL ENGINEERING: ENGINEERING AND THE BUILT ENVIRONMENT

ES216BB

ENGINEERING SOFTWARE DESIGN

**Evaluation 2
October 2024**

EVALUATION INSTRUCTIONS

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>.cpp” (e.g. **21011022.cpp**). 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.
5. **Evaluation Scope:** This assessment encompasses basic content from ES216AB and specifically ES216BB content defined in **Unit1 to Unit4**
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 given in the ".cpp" file is designed to manage a singly linked list where each node contains a string and a float value. The code starts by including necessary header files such as "iostream", "fstream", and "conio.h". 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 "Node" is defined to represent the nodes of the linked list. Each node contains a string ("sData"), a float ("fData"), and a pointer ("nextPtr") to the next node in the list. The code then includes the "EV3Memo.h" header file (change this to your header file name, "<studentnumber>.h"), which contains the function declarations for the functions that must be implemented.

The "main" function begins by initialising a pointer "startPtr" to "nullptr", indicating an empty linked list. It also declares variables "choice", "FileName", and "Average" to store the user's menu choice, the name of the file to be read, and the average of the float values in the linked list, respectively.

The program then enters a do-while loop, which displays a menu to the user with options to read and populate the linked list 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 all nodes 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 linked list. If the second option is chosen, the "**AverageLL**" 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 "**DisplayNodeData**" function to show the data in the linked list. The fifth option deletes all nodes in the linked list using the "**DeleteAllNodes**" 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.

FUNCTIONS

The required functions for the C++ program should be filled out in the provided ".h" file, which has been renamed to match your student number. To compile the entire program, use the "Build and Run" option on the ".cpp" file. Keep in mind that if any of the specified functions are missing declarations, the program won't successfully compile or run.

1. Insert Node Function

void InsertNode (Node **sPtr, string StringData, float FloatData);

This function is designed to insert a new node at the end of a linked list. It takes in a double pointer to the start of the list ("sPtr"), a string ("StringData"), and a float ("FloatData"). The function first creates a new node and assigns the string and float values to the node's respective data fields. It also sets the "nextPtr" of the new node to "nullptr", indicating that this will be the last node in the list. If the list is currently empty (i.e., the start pointer is "nullptr"), the new node becomes the first node. Otherwise, the function traverses the list to find the last node and appends the new node to the end.

2. Read Text File Function

void ReadTextFile (string FileName, Node **sPtr);

This function reads data from a text file and inserts it into a linked list. It takes a filename ("FileName") and a double pointer to the start of the list ("sPtr"). The function attempts to open the file and, if successful, reads pairs of string and float values from it. For each pair read, it calls the "InsertNode" function to add the data to the linked list. Once all data is read, the file is closed.

3. Average Linked List Function

float AverageLL (Node **sPtr);

This function calculates and returns the average of the float values stored in the linked list. It takes a double pointer to the start of the list ("sPtr"). If the list is empty, it returns 0. Otherwise, it traverses the list, summing up the float values and counting the number of nodes. Finally, it returns the average by dividing the sum by the count.

4. Display Node Data Function 1 - Overloaded

void DisplayNodeData (Node **sPtr);

This function displays the data of each node in the linked list. It takes a double pointer to the start of the list ("sPtr"). For each node, it prints out the string and float data. The function traverses the entire list, displaying data for each node until the end is reached.

5. Display Node Data Function 2 - Overloaded

void DisplayNodeData (Node **sPtr, float avg);

This version of the "DisplayNodeData" function is an extension of the previous one. In addition to displaying the string and float data of each node, it also calculates and displays the deviation of the float data from a given average ("avg"). The deviation is calculated as the difference between the float data of the node and the provided average.

6. Delete All Nodes Function

void DeleteAllNodes (Node **sPtr);

This function is responsible for deallocating the memory used by the linked list. It takes a double pointer to the start of the list ("sPtr"). The function traverses the list, deleting each node and moving to the next one. After all nodes are deleted, it sets the start pointer to "nullptr", indicating an empty list.

PROGRAM PRINT SCREENS

Home Screen Menu

```
-----  
|      Evaluation 3      |  
-----  
1. Read and Populate LL  
2. Calculate LL Average  
3. Display String Data  
4. Display Float Data  
5. Delete Nodes and Exit  
Choice:
```

Option 1 – Import Text File Data and Update Linked List

```
-----  
|      Evaluation 3      |  
-----  
1. Read and Populate LL  
2. Calculate LL Average  
3. Display String Data  
4. Display Float Data  
5. Delete Nodes and Exit  
Choice: 1  
  
File Name: InputData.txt  
  
Press any key to continue...
```

Option 2 – Calculate Linked List Float Data Average

```
-----  
|      Evaluation 3      |  
-----  
1. Read and Populate LL  
2. Calculate LL Average  
3. Display String Data  
4. Display Float Data  
5. Delete Nodes and Exit  
Choice: 2  
  
Average: 52.664  
Press any key to continue...
```

Option 3 – Display Linked List String and Float Data

```
-----  
|      Evaluation 3      |  
-----  
1. Read and Populate LL  
2. Calculate LL Average  
3. Display String Data  
4. Display Float Data  
5. Delete Nodes and Exit  
Choice: 3  
  
String Data : RoofSensor  
Float Data  : 41.99  
  
String Data : KitchenSensor  
Float Data  : 78.22  
  
String Data : BatterySensor  
Float Data  : 67  
  
String Data : GarageSensor  
Float Data  : 44.11  
  
String Data : OutdoorSensor  
Float Data  : 32  
  
Press any key to continue...
```


Option 4 – Display Linked List string and Deviation Data

```
-----  
|      Evaluation 3      |  
-----  
1. Read and Populate LL  
2. Calculate LL Average  
3. Display String Data  
4. Display Float Data  
5. Delete Nodes and Exit  
Choice: 4  
  
String Data      : RoofSensor  
Float Data       : 41.99  
Average Deviation : -10.674  
  
String Data      : KitchenSensor  
Float Data       : 78.22  
Average Deviation : 25.556  
  
String Data      : BatterySensor  
Float Data       : 67  
Average Deviation : 14.336  
  
String Data      : GarageSensor  
Float Data       : 44.11  
Average Deviation : -8.554  
  
String Data      : OutdoorSensor  
Float Data       : 32  
Average Deviation : -20.664  
  
Press any key to continue...
```

Option 5 – Exit

```
C:\Users\Antonie\Desktop\Evaluation 3\EV3Memo.e  
-----  
|      Evaluation 3      |  
-----  
1. Read and Populate LL  
2. Calculate LL Average  
3. Display String Data  
4. Display Float Data  
5. Delete Nodes and Exit  
Choice: 5  
  
Exit Program...
```

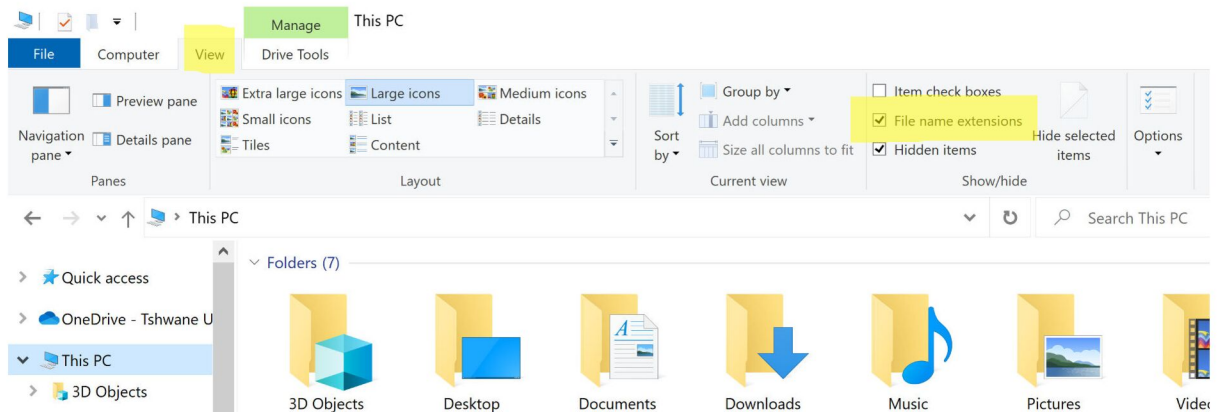

Text File Content Print Screen – InputData.txt



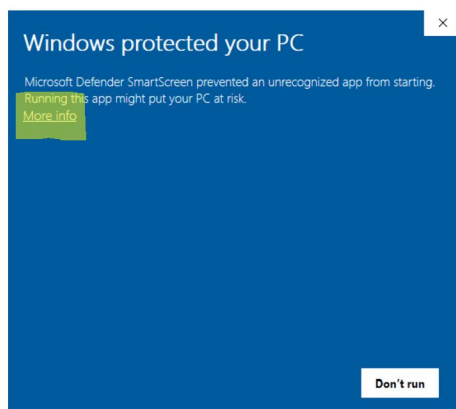
```
InputData.txt - Notepad
File Edit Format View Help
RoofSensor      41.99
KitchenSensor   78.22
BatterySensor    67
GarageSensor     44.11
OutdoorSensor    32
|
Ln 6, Col 1    100%    Windows (CRLF)    UTF-8
```

HOW TO RUN THE SHOWCASE FILE

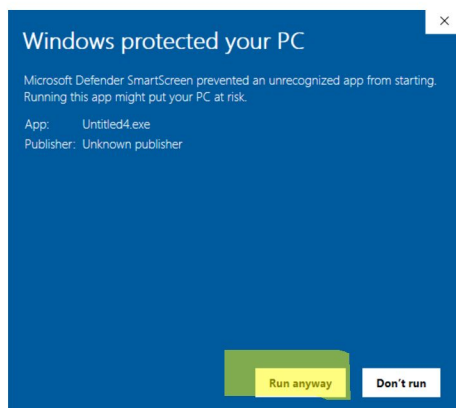
1. Enable file extensions (see highlighted in yellow)



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**”



5. Click on “**Run anyway**”



ANNEXURE A – MARK ALLOCATION

TEST RUBRIC	WEIGHT [%]	MARK [%]
C CODE & OUTPUT EVALUATION	50	0
1. Insert Node Function (Insert new node into LL)	8	0
2. Read Text File Function (Read string and float pairs from Text File)	8	0
3. Average Linked List Function (Calculate LL float data average)	6	0
4. Display Node Data Function 1 (Display LL string and float data)	6	0
5. Display Node Data Function 2 (Display LL string and average deviation)	6	0
6. Delete All Nodes Function	6	
7. Overall Impression (Neatness, Readability, Spacing and Indentation)	5	
8. No Compile or Runtime errors	5	0
TOTAL	50	0

Graduate Attribute	GA Number	GA Score [0-5]
Application of scientific and engineering knowledge	GA2	3 and 5
Engineering methods, skills, tools, including information technology	GA5	1,2,4, and 6
Impact of Engineering Activity	GA7	8
Engineering Professionalism	GA10	7

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: Dereference: * 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	 	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		DEL

Source: www.LookupTables.com