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

Mr D. Engelbrecht

MODERATOR:

Mr TE Olivier

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 2025

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 your 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. 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.

5. **Evaluation Scope:** This assessment encompasses basic content from ES216AB and specifically ES216BB content defined in **Units 1 to 3**

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 template ".cpp" file except for implementing the requested functions as required.
    c. Use the exact function names and parameters as used in the evaluation question paper and template ".cpp" file.
    d. Complete the C++ functions below the main function in each comment block as shown.

C++ FILE CODE EXPLANATION

The provided C++ code manages a dynamically created array of inventory items, where each record consists of an item's ID (integer), name (string), quantity (integer), and unit price (float).

STRUCTURE DEFINITION

A struct named InventoryItem is defined with the following members:

- *int itemID*
- *string itemName*
- *int quantity*
- *float unitPrice*

MAIN FUNCTION DESCRIPTION

The main function initialises a pointer for the dynamically allocated array named inventory, initially set to **nullptr**. Variables such as **arraySize**, **userChoice**, **fileName**, and **totalInventoryValue** are declared to manage array size, user menu selection, file input, and total inventory value, respectively.

The program presents a menu-driven interface to the user, providing these options:

1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items (quantity below threshold)
5. Delete inventory data and exit

Based on the user's choice, the corresponding functions are executed through a switch-case statement.

FUNCTIONS IMPLEMENTATION

---

1. InitialiseInventoryItem Function

void InventoryItem::InitialiseInventoryItem(int id, string name, int qty, float price);

- **Purpose:** Initialises individual members of an inventory item record.
- **Parameters:**
  - *id* - Integer representing the item's identification number.
  - *name* - String representing the item's name.
  - *qty* - Integer representing the quantity in stock.
  - *price* - Float representing the unit price of the item.
- **Returns:** No return value; it initialises the structure's members.

---

2. TextFileLineCount Function

int TextFileLineCount(string fileName);

- **Purpose:** Counts lines in a text file, each line representing one inventory item.
- **Parameters:**
  - *fileName* - Name of the file (with extension) to read from.
- **Returns:** Integer value representing the total number of lines (inventory items).

---

3. ReadFileAndPopulate Function

void ReadFileAndPopulate(string fileName, InventoryItem **inventory, int *arraySize);

- **Purpose:** Reads inventory data from a file and populates the dynamically allocated array of inventory items.
- **Parameters:**
  - *fileName* - Name of the file to read data from.
  - *inventory* - Double pointer to the array of InventoryItem structures to populate.
  - *arraySize* - Pointer to an integer to store the number of inventory items.
- **Returns:** No return value; populates the array and updates the array size.

---

4. CalculateTotalInventoryValue Function

float CalculateTotalInventoryValue(InventoryItem *inventory, int arraySize);

- **Purpose:** Calculates the total value of all inventory items.
- **Parameters:**
  - *inventory* - Pointer to the array of inventory items.
  - *arraySize* - Number of inventory items in the array.
- **Returns:** Float value representing the total inventory value.

---

5. DisplayAllInventoryItems Function

void DisplayAllInventoryItems(InventoryItem *inventory, int arraySize);

- **Purpose:** Displays all inventory items in a structured table.
- **Parameters:**
  - *inventory* - Pointer to the array of inventory items.
  - *arraySize* - Number of inventory items in the array.
- **Returns:** No return value; prints data directly to the console.

---

6. DisplayLowStockItems Function

void DisplayLowStockItems(InventoryItem *inventory, int arraySize, int threshold);

- **Purpose:** Displays inventory items with quantities below a specified threshold.
- **Parameters:**
  - *inventory* - Pointer to the array of inventory items.
  - *arraySize* - Number of inventory items in the array.
  - *threshold* - Quantity threshold to identify low-stock items.
- **Returns:** No return value; prints data directly to the console.

---

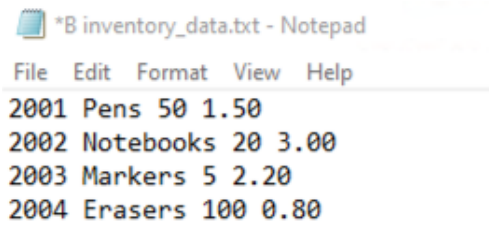
7. DeleteInventoryArray Function

void DeleteInventoryArray(InventoryItem **inventory, int *arraySize);

- **Purpose:** Deallocates memory allocated for the inventory array and resets pointers and array size.
- **Parameters:**
  - *inventory* - Double pointer to the array of inventory items.
  - *arraySize* - Pointer to the integer representing the number of inventory items.
- **Returns:** No return value; resets pointers and array size to initial states.

PRINT SCREENS

Text File Content:



Main Menu:

```
-----------------------------------
     Inventory Management System
-----------------------------------
1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items
5. Delete data and exit
Choice:
```

Load text file:

```
-----------------------------------
     Inventory Management System
-----------------------------------
1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items
5. Delete data and exit
Choice: 1

File Name: inventory_data.txt

Press any key to continue...
```

Calculate total inventory value:

```
-----------------------------------
     Inventory Management System
-----------------------------------
1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items
5. Delete data and exit
Choice: 2

Total Inventory Value: $226

Press any key to continue...
```

Display all inventory items:

```
-----------------------------------
     Inventory Management System
-----------------------------------
1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items
5. Delete data and exit
Choice: 3
-----------------------------------------------------
        ID           Name       Qty     Price
-----------------------------------------------------
      2001           Pens        50       1.5
      2002      Notebooks        20         3
      2003        Markers         5       2.2
      2004        Erasers       100       0.8
-----------------------------------------------------

Press any key to continue...
```

Display low-stock items:

```
-----------------------------------
     Inventory Management System
-----------------------------------
1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items
5. Delete data and exit
Choice: 4

Enter threshold quantity: 25
Items below quantity threshold (25):
2002: Notebooks - 20
2003: Markers - 5

Press any key to continue...
```

Delete data and exit:

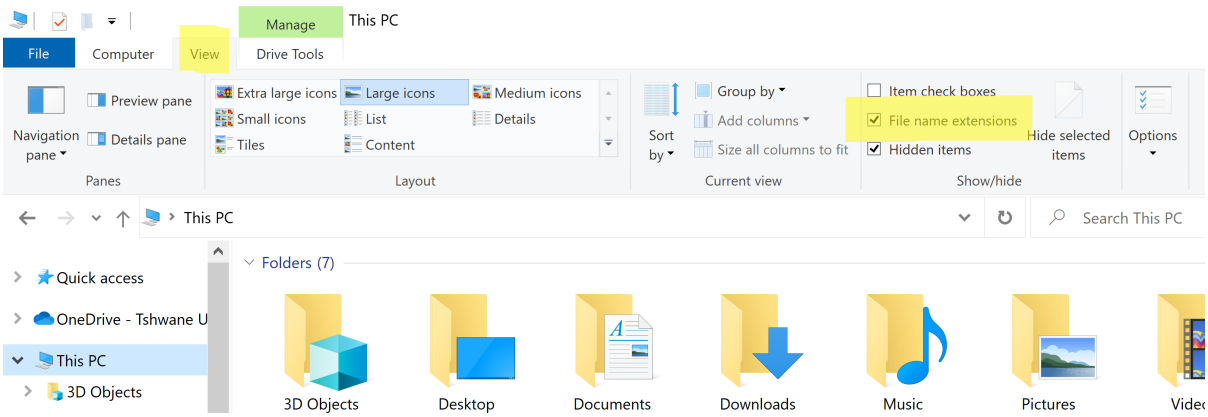
```
-----------------------------------
     Inventory Management System
-----------------------------------
1. Load inventory data from file
2. Calculate total inventory value
3. Display all inventory items
4. Display low-stock items
5. Delete data and exit
Choice: 5

Exiting program...

Process returned 0 (0x0)   execution time : 320.782 s
Press any key to continue.
```

HOW TO RUN THE SHOWCASE FILE

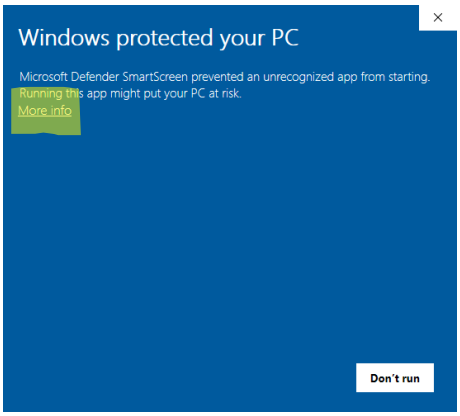
1. Enable file extensions (see highlighted in yellow)



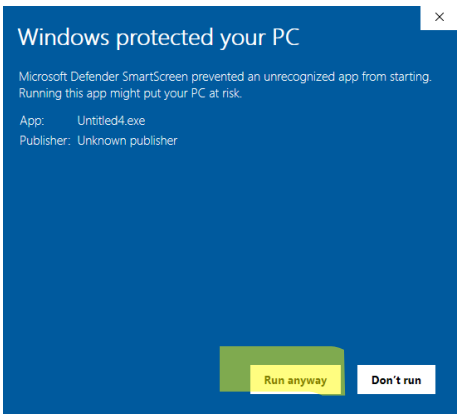
2. Change the name from **"Showcase.old"** to **"Showcase.exe"**

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

4. Windows may show the following. Click on **"More info"**



5. Click on **"Run anyway"**



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		**50+2**
1. InitialiseInventoryItem Function (Initialize structure members with default parameters)		**5**
2. TextFileLineCount Function (Count data lines in text file to determine array size)		**5**
3. Read and Populate Function **3.1 Deallocate Memory Before Population**		**2**
3. Read and Populate Function **3.2 Retrieve Line Count**		**2**
3. Read and Populate Function **3.3 Allocate Dynamic Memory & Set Size**		**2**
3. Read and Populate Function **3.4 Check Memory Allocation Failure**		**2**
3. Read and Populate Function **3.5 Populate Data from File**		**4**
4. CalculateTotalInventoryValue Function (Calculate total inventory value)		**5**
5. DisplayAllInventoryItems Function (Display array data)		**5**
6. DisplayLowStockItems Function (Display correct array data)		**5**
7. DeleteInventoryArray Function (Deallocate dynamic array memory)		**5**
8. Overall Impression (Neatness, Readability, Spacing, and Indentation)		**5**
9. Compile & Runtime Stability		**5**
TOTAL		**50**

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) { … };

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

Source: www.LookupTables.com