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:

120 Minutes

FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT

DEPARTMENT OF ELECTRICAL ENGINEERING

ES216BB ENGINEERING SOFTWARE DESIGN B

EVALUATION 3

NOVEMBER 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 "**.h**" 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 **Unit1 to Unit5**

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:**

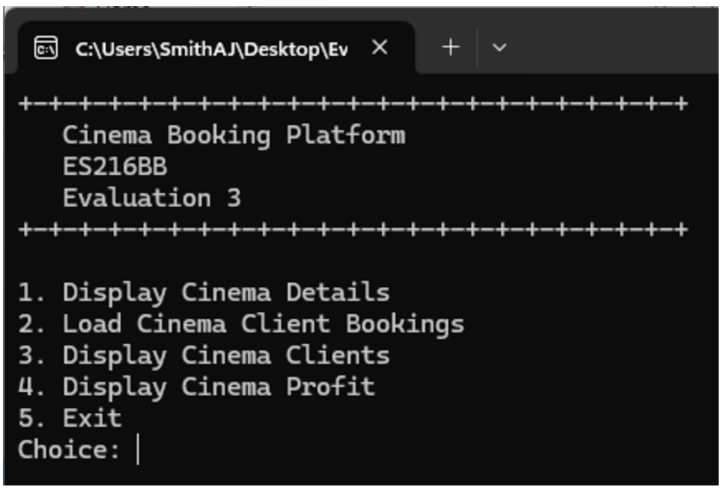
    a. Remember to save your work on the PC "D: Drive" and save regularly throughout the evaluation.
    b. Do not modify the given code in the "**.cpp**" file except for the include statement of your own header file.
    c. Use the exact function names and parameters as used in the in the "**.cpp**" file and as defined in the question paper.
    d. Complete the C++ class definition and class functions inside the designated areas as indicated in the header file template.

C++ FILE CODE EXPLANATION

You will be provided with a C++ file that sets up a basic program to manage a cinema booking system. Your task will involve understanding how the main program interacts with a class, CinemaBooking, defined in the header file, which you'll need to create based on this .cpp file's structure.

The main function is organised into a simple menu-based interface, allowing users to:

- **Display Cinema Details**: Shows information about the cinema, like its name, the movie being screened, and maximum seating capacity.
- **Load Client Bookings**: Reads client names from a file, Bookings.txt, and adds each client to a booking list in the CinemaBooking class.
- **Display Cinema Clients**: Displays a list of all clients who have made bookings for the movie.
- **Display Cinema Profit**: Calculates and displays the profit based on the ticket price and the number of booked seats.



This .cpp file is designed to work with functions defined in the CinemaBooking class, and your task will include creating this class and ensuring it includes necessary functions like SetDetails, AddClient, DisplayCinemaDetails, DisplayClientList, and CinemaProfit. The basic structure and logic in the .cpp file demonstrate how these functions are intended to interact with the main program.

CLASS DEFINITION AND FUNCTIONS

The required class definition and functions for the C++ header file should be implemented in the appropriate comment blocks as given in the **.h** template file. The class and function declarations and descriptions are as follows:

0. CinemaBooking Class Definition

The CinemaBooking class represents a cinema booking system, managing cinema and movie details, a list of client bookings, and calculating cinema profits. The class contains both private and public members:

- **Private Members**:
  - **CinemaName**: Stores the name of the cinema.
  - **MovieName**: Stores the name of the movie being shown.
  - **MaxSeats**: Represents the maximum number of seats available.
  - **ClientNameList**: A dynamically allocated array of strings that holds the names of booked clients.
  - **SeatCount**: Keeps track of the current number of seats booked.

- **Public Functions**:
  - **CinemaBooking()**: Constructor to initialise the attributes.
  - **~CinemaBooking()**: Destructor to manage dynamic memory.
  - **SetDetails()**: Sets the cinema and movie details.
  - **AddClient()**: Adds a client to the booking list if seats are available.
  - **DisplayCinemaDetails()**: Displays details about the cinema and the movie.
  - **DisplayClientList()**: Shows a list of all clients booked.
  - **CinemaProfit()**: Calculates and returns the total profit based on the ticket price.

1. Constructor

CinemaBooking::CinemaBooking()

The constructor function initialises the CinemaBooking class with default values:

- Sets CinemaName and MovieName to empty strings.
- Initializes MaxSeats and SeatCount to 0.
- Sets ClientNameList to nullptr as no clients are initially booked.

Key Operations:

1. **Default Initialization**: All member variables are set to default values, ensuring the class starts in a consistent state.
2. **Memory Safety**: ClientNameList is set to nullptr, avoiding accidental access to uninitialised memory.

2. Destructor

CinemaBooking::~CinemaBooking()

The destructor function handles the cleanup of dynamically allocated memory.

- Deletes the ClientNameList array when the object is destroyed, freeing any allocated memory to avoid memory leaks.

Key Operations:

1. **Memory Management**: Ensures that the ClientNameList array is properly deallocated when the object is destroyed.

3. SetDetails Function

void CinemaBooking::SetDetails(string CN, string MN, int MS)

The SetDetails function sets basic information about the cinema:

- CN (Cinema Name), MN (Movie Name), and MS (Maximum Seats) are used to initialise CinemaName, MovieName, and MaxSeats.

Key Operations:

1. **Attribute Setting**: Updates cinema and movie details as well as maximum seating capacity.

4. AddClient Function

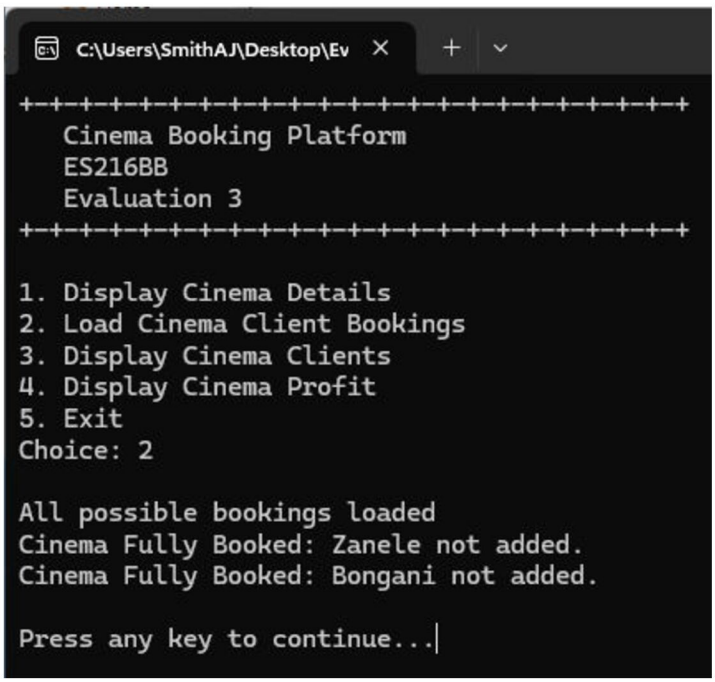
void CinemaBooking::AddClient(string ClientName)

The AddClient function adds a client to the cinema's booking list if seats are available:

- Checks if SeatCount equals MaxSeats. If true, the cinema is fully booked, and the client is not added.
- If ClientNameList is nullptr, it creates an array for the first client.
- For additional clients:
  - Copies the existing client names into a temporary array.
  - Deletes the old ClientNameList and creates a new array with one extra slot.
  - Copies back the old clients and adds the new client.
  - Deletes the temporary array.

Key Operations:

1. **Seat Availability Check**: Ensures clients are only added if seats are available.
2. **Dynamic Memory Allocation**: Manages the resizing of ClientNameList array safely to accommodate more clients.
3. **Array Resizing and Copying**: Utilizes a temporary copy to expand and repopulate the client list with the new client.



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   Cinema Booking Platform
   ES216BB
   Evaluation 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

1. Display Cinema Details
2. Load Cinema Client Bookings
3. Display Cinema Clients
4. Display Cinema Profit
5. Exit
Choice: 2

All possible bookings loaded
Cinema Fully Booked: Zanele not added.
Cinema Fully Booked: Bongani not added.

Press any key to continue...
```

5. DisplayCinemaDetails Function

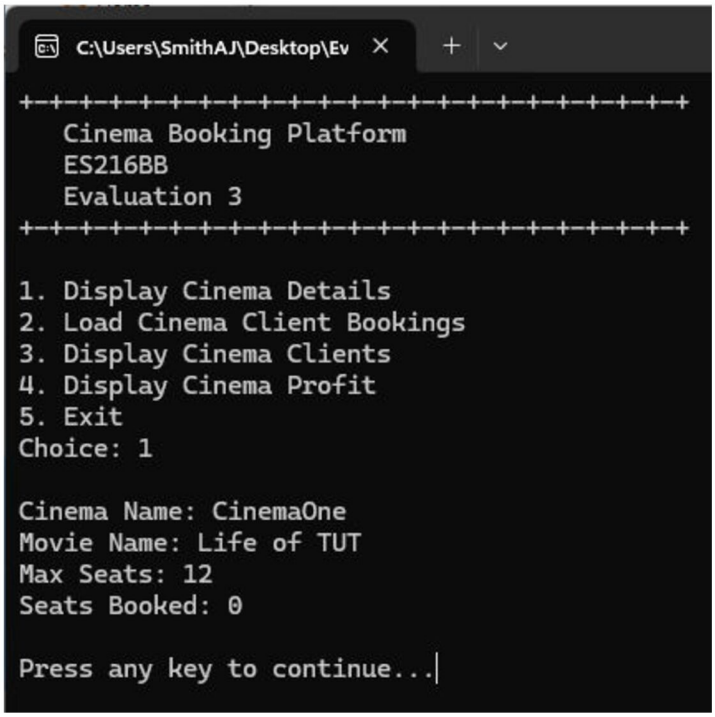
void CinemaBooking::DisplayCinemaDetails()

The DisplayCinemaDetails function displays details about the cinema and movie:

- Prints the cinema name, movie name, maximum seat capacity, and current seat count.

Key Operations:

1. **Display Information**: Outputs basic cinema details, providing context on bookings.



6. DisplayClientList Function

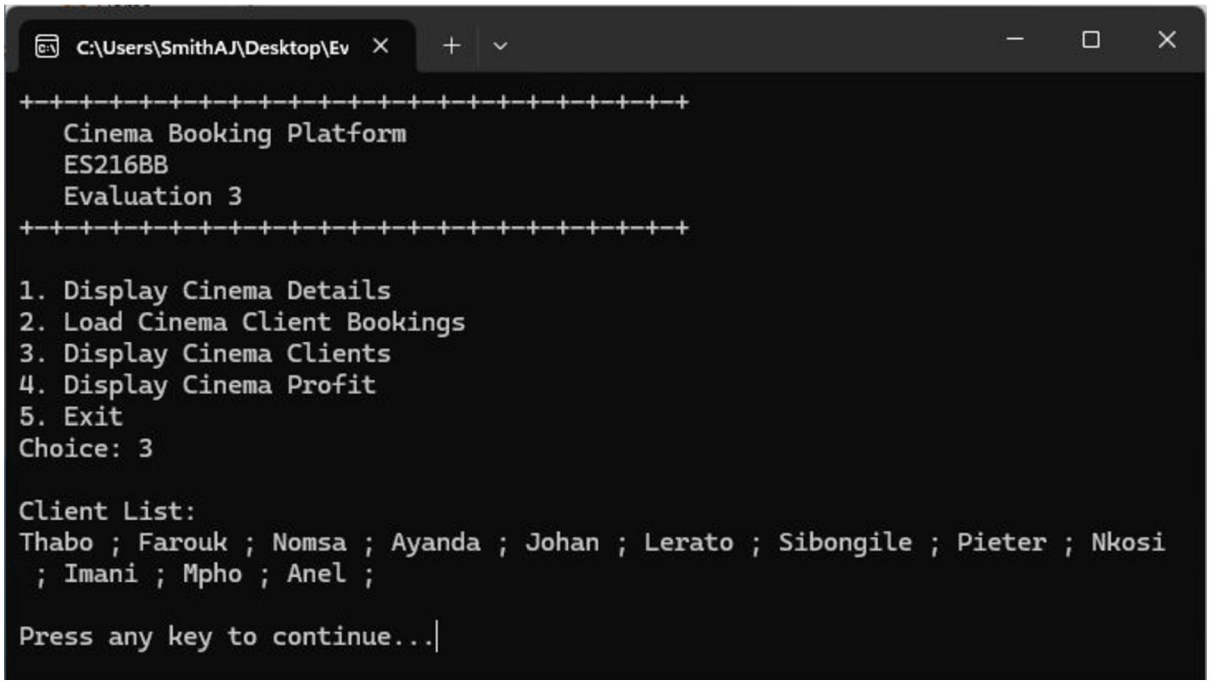
void CinemaBooking::DisplayClientList()

The DisplayClientList function displays a list of clients who have booked seats:

- Check if SeatCount is zero. If so, it indicates that no clients are booked.
- Otherwise, iterates through ClientNameList and prints each client name.

Key Operations:

1. **Conditional Display**: Outputs a message if no clients are booked.
2. **Client List Display**: Iterates through and displays each client's name.



7. CinemaProfit Function

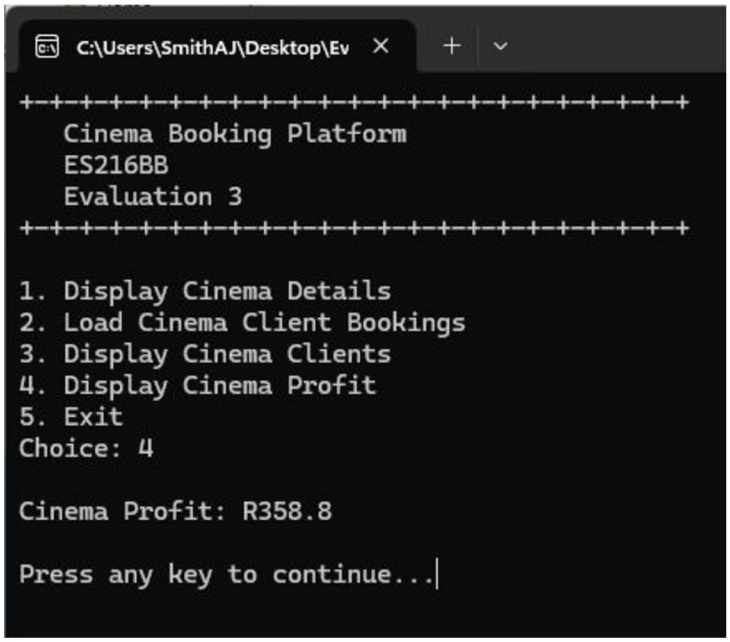
float CinemaBooking::CinemaProfit(float TicketPrice)

The CinemaProfit function calculates the total profit for the cinema based on ticket sales:

- Multiplies SeatCount by TicketPrice to compute total revenue from bookings.

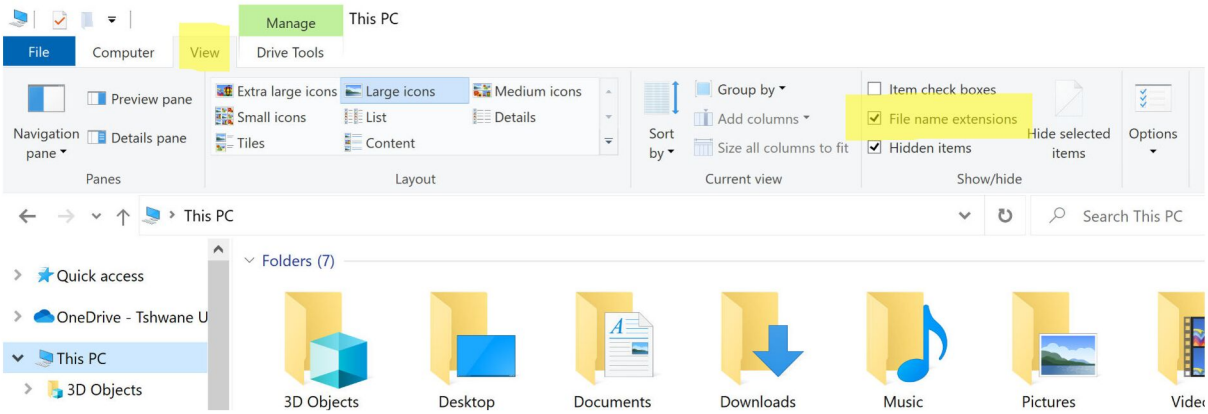
Key Operations:

1. **Profit Calculation**: Returns the profit by multiplying the number of booked seats by the ticket price.



HOW TO RUN THE SHOWCASE FILE

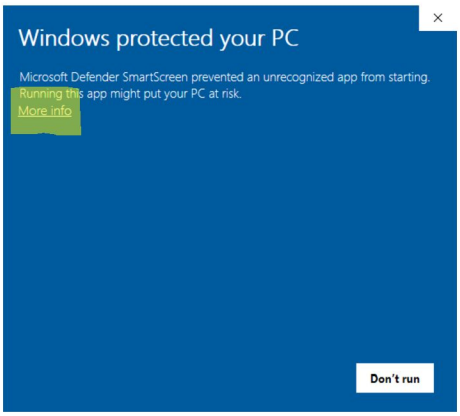
1. Enable file extensions (see highlighted in yellow)



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

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**
0. Class Definition (Initialise class definition with specified members)		**7**
1.Class Constructor Function (Initialise class variables in the function)		**5**
2. Class Destructor Function (Delete all created dynamic memory in the function)		**5**
3. Set Details Function (Initialise class variables with received parameters)		**5**
4. Add Function (With each function call grow dynamic array accordingly)		**10**
5. Display Details Function (Dispay object specific details)		**5**
6. Display List Function (Display all dynamically added data)		**5**
7. Profit Function (Display profit as calculated by amount of booking)		**3**
8. Overall Impression (Neatness, Readability, Spacing, and Indentation)		**5**
9. No Compile or Runtime errors		**5**
TOTAL		**50**
STUDENT NUMBER		

Graduate Attribute	GA Number	GA Score [0-5]
Application of scientific and engineering knowledge	GA2	4,7
Engineering methods, skills, tools, including information technology	GA5	0,1,2,3
Impact of Engineering Activity	GA7	5,6,9
Engineering Professionalism	GA10	8,9

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		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