



**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

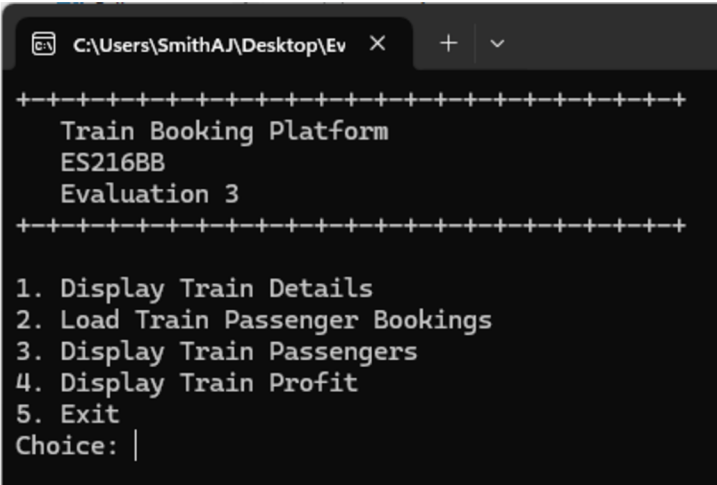
1. **Plagiarism:** Submit only original work. We will use similarity software to verify the authenticity of all submissions.
2. **Permitted Tools:** You are allowed to use only **CodeBlocks**, and **Google Chrome** to access the evaluation, view the evaluation PDF and upload submission for this evaluation. 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

This C++ file sets up a train booking management program. The program uses a menu system to interact with the TrainBooking class, which stores details about the train and passengers and calculates profits. You will need to use this structure to create the header file, which defines the necessary functions and class members.

The main function provides a menu interface for:

- **Displaying Train Details:** Shows details like train number, destination, and maximum seating.
- **Loading Passenger Bookings:** Reads passenger names from Bookings.txt and adds them to the list of passengers in the TrainBooking class.
- **Displaying Passenger List:** This shows the list of passengers booked on the train.
- **Displaying Train Profit:** Calculates the total revenue based on ticket prices and the number of bookings.



```
C:\Users\SmithAJ\Desktop\Ev X + v
+++++
Train Booking Platform
ES216BB
Evaluation 3
+++++

1. Display Train Details
2. Load Train Passenger Bookings
3. Display Train Passengers
4. Display Train Profit
5. Exit
Choice: |
```

This .cpp file provides the framework for interacting with the TrainBooking class. You will need to define functions such as SetDetails, AddPassenger, DisplayTrainDetails, DisplayPassengerList, and TrainProfit in the TrainBooking class to make the program fully functional.

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. TrainBooking Class Definition

The TrainBooking class represents a train booking system that manages train details and a list of booked passengers and calculates the profit from ticket sales. The class includes both private and public members:

- **Private Members:**
 - TrainNumber: Holds the identification number for the train.
 - DestinationName: Stores the name of the train's destination.
 - MaxSeats: Indicates the maximum number of available seats.
 - PassengerNameList: A dynamically allocated array for storing passenger names.
 - SeatCount: Keeps track of the number of seats that have been booked.
- **Public Functions:**
 - TrainBooking(): Constructor to initialize class members with default values.
 - ~TrainBooking(): Destructor to handle memory deallocation.
 - SetDetails(): Sets the train number, destination, and maximum seat capacity.
 - AddPassenger(): Adds a passenger to the booking list if seats are available.
 - DisplayTrainDetails(): Outputs details about the train and its destination.
 - DisplayPassengerList(): Displays a list of all booked passengers.
 - TrainProfit(): Calculates the profit based on the ticket price.

1. Constructor

TrainBooking::TrainBooking()

The constructor initialises the TrainBooking class with default values:

- Sets TrainNumber and DestinationName to empty strings.
- Initializes MaxSeats and SeatCount to 0.
- Sets PassengerNameList to nullptr, indicating no passengers are initially booked.

Key Operations:

1. **Default Initialization:** Ensures all member variables have default values, providing a consistent starting state.
2. **Memory Safety:** Setting PassengerNameList to nullptr prevents unintended access to uninitialised memory.

2. Destructor

TrainBooking::~~TrainBooking()

The destructor manages memory cleanup for PassengerNameList:

- Deletes the PassengerNameList array when the object is destroyed, freeing memory and preventing memory leaks.

Key Operations:

1. **Memory Management:** Ensures that the dynamically allocated array for PassengerNameList is deallocated when the object is destroyed.

3. SetDetails Function

void TrainBooking::SetDetails(string TN, string DN, int MS)

The SetDetails function initializes the train-specific details:

- Sets TrainNumber, DestinationName, and MaxSeats based on the provided values TN (Train Number), DN (Destination), and MS (Maximum Seats).

Key Operations:

1. **Attribute Initialization:** Updates the details for the train, including train number, destination, and seating capacity.

4. AddPassenger Function

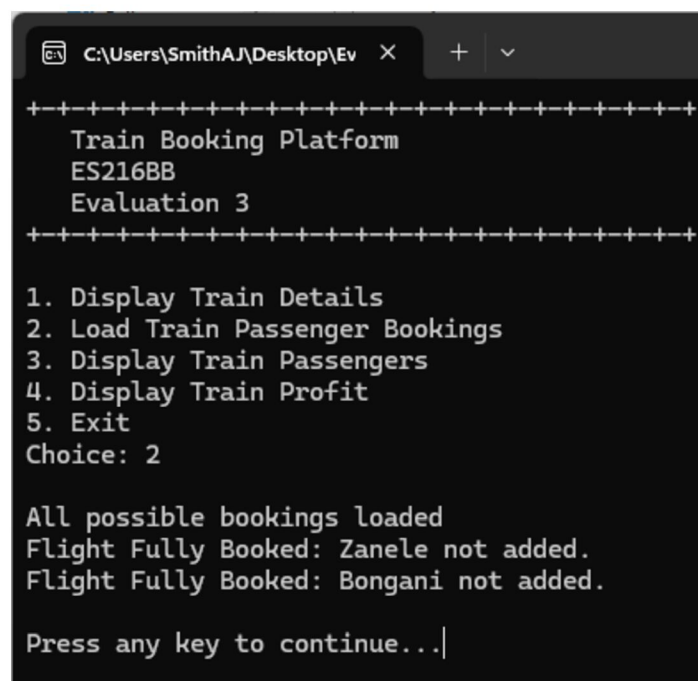
void TrainBooking::AddPassenger(string PassengerName)

The AddPassenger function adds a passenger to the booking list if there is availability:

- Check if SeatCount equals MaxSeats. If true, the train is fully booked, and the passenger is not added.
- If PassengerNameList is nullptr, initialize a new array to store the first passenger.
- For additional passengers:
 - Copies the existing passenger names into a temporary array.
 - Deletes the old PassengerNameList and creates a new array with one additional slot.
 - Copies back the existing passengers adds the new passenger and deletes the temporary array.

Key Operations:

1. **Seat Availability Check:** Ensures passengers are only added if seats are available.
2. **Dynamic Memory Allocation:** Manages the resizing of PassengerNameList to accommodate new passengers.
3. **Array Expansion and Repopulation:** Uses a temporary copy to expand and repopulate the passenger list with new entries.



```
C:\Users\SmithAJ\Desktop\Ev X + v
+++++
Train Booking Platform
ES216BB
Evaluation 3
+++++

1. Display Train Details
2. Load Train Passenger Bookings
3. Display Train Passengers
4. Display Train Profit
5. Exit
Choice: 2

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

Press any key to continue...|
```

5. DisplayTrainDetails Function

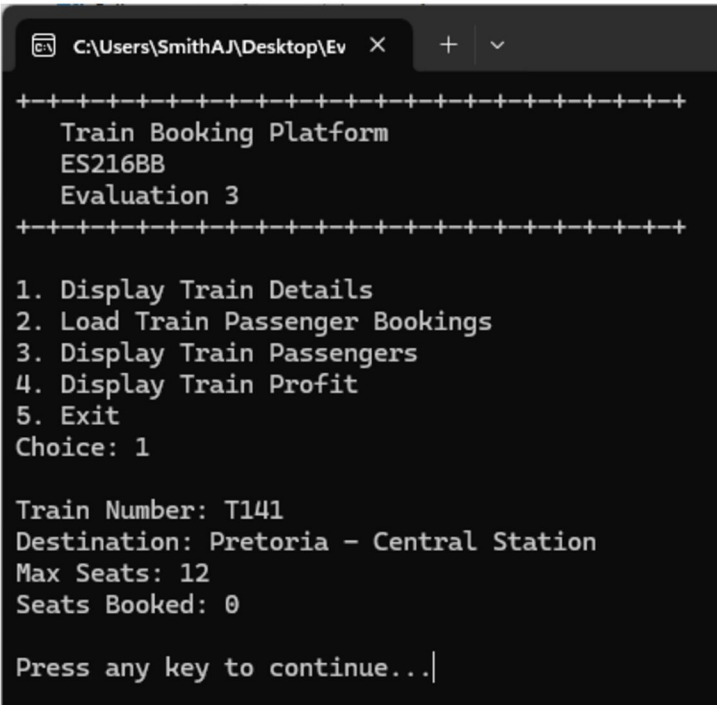
void TrainBooking::DisplayTrainDetails()

The DisplayTrainDetails function outputs information about the train:

- Prints the train number, destination, maximum seat capacity, and the number of booked seats.

Key Operations:

1. **Information Display:** Output essential details about the train and its booking status.



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

1. Display Train Details
2. Load Train Passenger Bookings
3. Display Train Passengers
4. Display Train Profit
5. Exit
Choice: 1

Train Number: T141
Destination: Pretoria - Central Station
Max Seats: 12
Seats Booked: 0

Press any key to continue...|
```

6. DisplayPassengerList Function

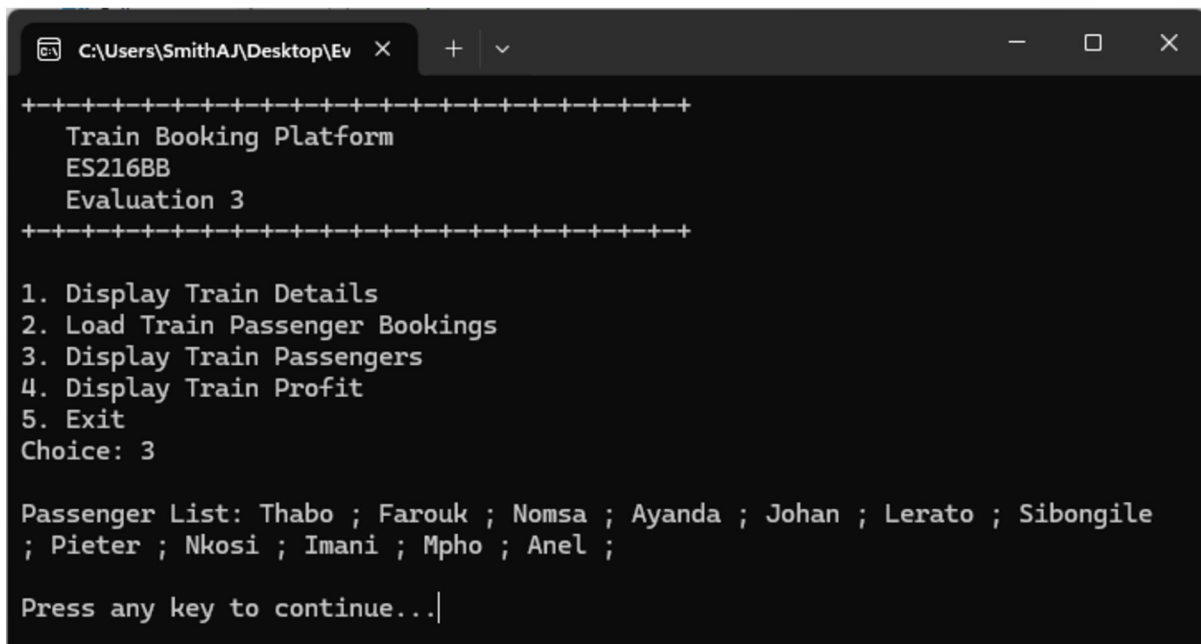
void TrainBooking::DisplayPassengerList()

The DisplayPassengerList function displays a list of booked passengers:

- Check if SeatCount is zero. If true, it displays a message indicating that no passengers are booked.
- Otherwise, iterates through PassengerNameList and prints each passenger's name.

Key Operations:

1. **Conditional Display:** Shows a message if no passengers are booked.
2. **List Display:** Iterates through and displays each booked passenger's name.



```
C:\Users\SmithAJ\Desktop\Ev X + - □ X
+++++
Train Booking Platform
ES216BB
Evaluation 3
+++++

1. Display Train Details
2. Load Train Passenger Bookings
3. Display Train Passengers
4. Display Train Profit
5. Exit
Choice: 3

Passenger List: Thabo ; Farouk ; Nomsa ; Ayanda ; Johan ; Lerato ; Sibongile
; Pieter ; Nkosi ; Imani ; Mpho ; Anel ;

Press any key to continue...|
```


7. TrainProfit Function

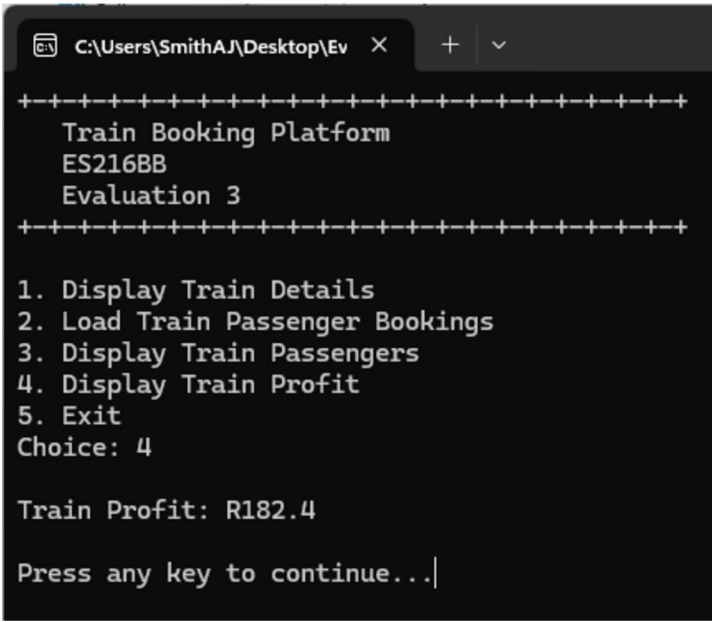
float TrainBooking::TrainProfit(float TicketPrice)

The TrainProfit function calculates the total revenue from ticket sales:

- Multiplies SeatCount by TicketPrice to compute the total revenue from booked seats.

Key Operations:

1. **Profit Calculation:** Calculates and returns the total revenue based on ticket price and number of booked seats.



```
C:\Users\SmithAJ\Desktop\Ev X + v
+-----+
Train Booking Platform
ES216BB
Evaluation 3
+-----+

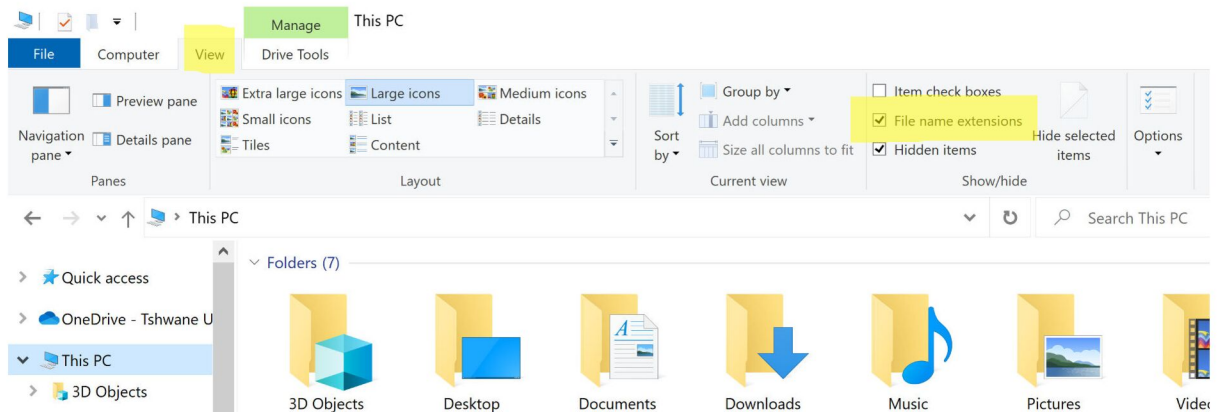
1. Display Train Details
2. Load Train Passenger Bookings
3. Display Train Passengers
4. Display Train Profit
5. Exit
Choice: 4

Train Profit: R182.4

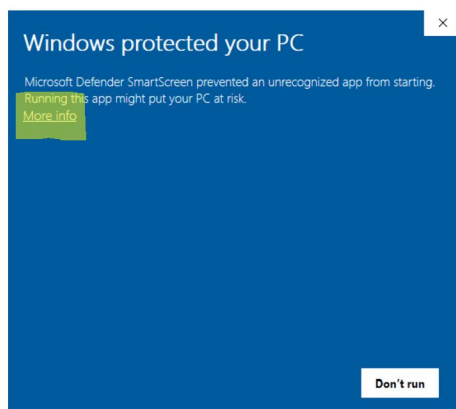
Press any key to continue...|
```

HOW TO RUN THE SHOWCASE FILE

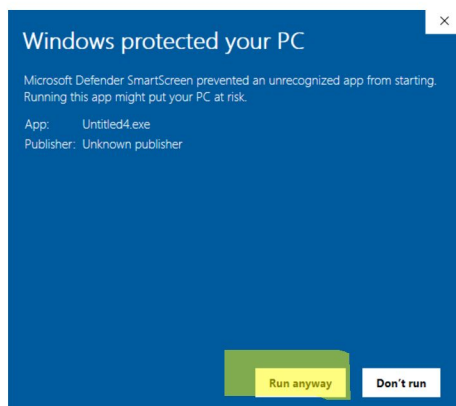
1. Enable file extensions (see highlighted in yellow)



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**”



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 (Display 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: 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