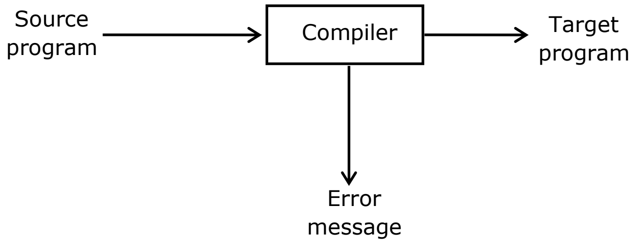
Abhishek Sharma C language notes

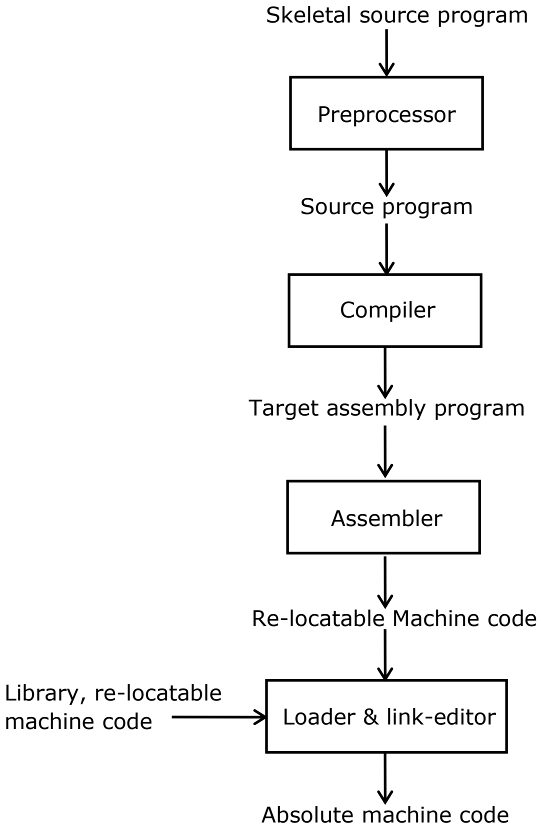
Lecture 1 Language processor

A compiler is a program that reads a program written in one language the source language and translates it into an equivalent program in another language the target language



A compiler

Language processing system:



Cousins of the compiler:

We will discuss the context in which a compiler typically operates

- Preprocessor
- Assembler
- Loader and Linker

Preprocessor:

Preprocessor produce input to compiler.

They may perform following function.

1. **Macro processing:** A preprocessor may allow a user to define macro that are shorthands for longer constructs.
2. **File inclusion:** A preprocessor may include header file into the program text. For example C processor causes, the context of the file <global.h> to replace the statement # include<global.h> when it processes a file, containing this statement.
3. **Rational preprocessor:** These processor augment older languages with more morden flow of control and data structuring facilities for example, such a preprocessor might provide user with built in macros for construct like while statement or if-statements, where none exists in the programming language itself.
4. **Language Extensions:** These processors attempt to add capabilities to the language by what amounts to built in macros

Loader and link editors: A program called a loader performs loading and link editing. The process of loading consists of taking re-locatable machine code altering the re-locatable addresses and placing the altered instruction and data in memory at the proper location.

Link editor allows us to make a single program from several files of relocatable machine code and planning the altered instruction and data in memory at the proper location.

The link editor allows us to make a single program from several files of relocatable machine code. These files may have been the result of several different compilations and one or may be library files of routine provided by the system and available to any program needs then.

Assembler: Some compiler produces assembly code that's passed to an assembler for further processing. Other compilers perform the job of the assembler, producing relocatable machine code that can be passed directly to loader/link editors. Assembly code is mnemonic version of machine code in which names are used instead of binary codes for operation and names are also given to memory addresses. A typical sequence of assembly instruction might be

$\text{MOV}\ \ a, R_1$

$\text{ADD}\ \ \#2, R_1$

$\text{MOV}\ \ R_1, b$

This code moves the contents of the address a into register1, then adds the constant 2 to it treating the content of register1 as a fixed point number and finally stores the result in the location named by b thus it computes b=a+2

Integer Representation: Lecture 2 and 3

int a = 10;

C Language Declaration

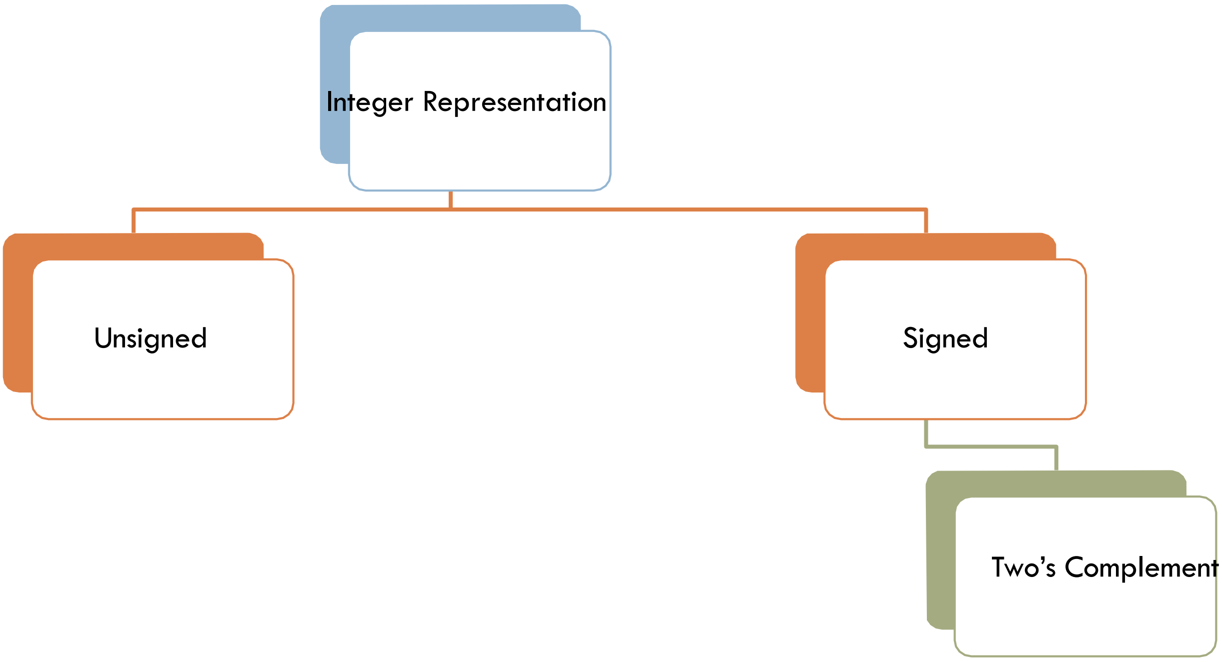
Integer Number: is a whole number without fractions, it can be positive or negative

Integers range between negative infinity

(– ∞) and positive infinity (+ ∞)

But can a computer store all the integers in between?

Integer Representation in C



Unsigned Integer

Unsigned Integer: is an integer without a sign and ranges between 0 and + ∞

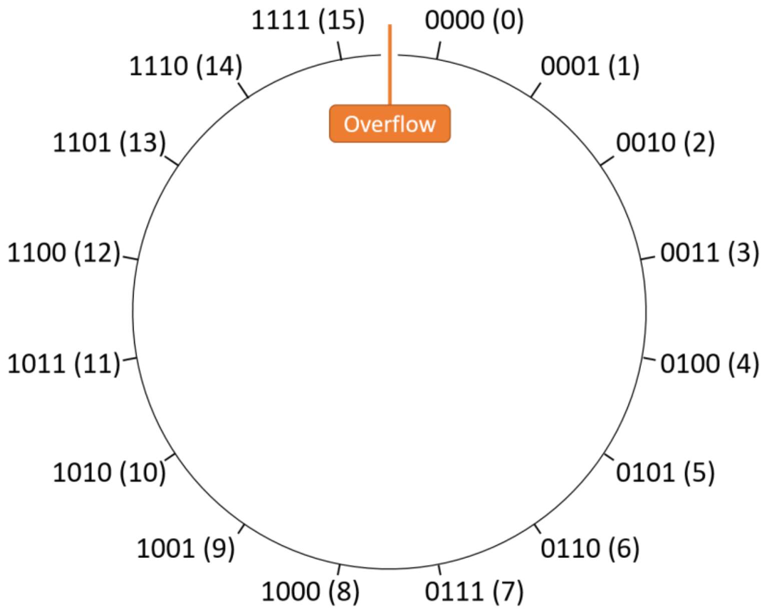
The maximum unsigned integer depends on the number of bits (N) allocated to represent the unsigned integer in a computer

0 to (2^{N-1})

No. of bits	Range		
8	0	to	255
16	0	to	65535
32	0	to	?

Over flow

if we try store value that's not in the range then overflow occurs



Suppose number is 10 , is the number is in range
Answer is yes

Suppose number is 20 , is the number is in range , answer is no Hence overflow occurs.

If over flow occurs how to find the value

If n bit number then and given number is x then unsigned number printed by the program is

$$\textbf{X mod } \mathbf{2^n}$$

Lecture 3, 4 Problem practice

Example 1

If the number is 20 and n is 4 bits what is the unsigned value printed?
(A) 20
(B) -20
(C) 4
(D) -4

Answer
4
Explanation :

$20 \bmod 2^4 = 4$

Example 2

If the number is 128 and n is 8 bits what is the unsigned value printed?
(A) 128
(B) -128
(C) 64

(D) -64

Answer

128

Explanation

128 mod 2^8 = 128 mod256= 128

Example 3

What is the range of 8 bit unsigned Integer

A. 0 to 256
B. 1 to 256
C. 0 to 255
D. 1 to 255

Answer

C

Example 4

What is the range of 7 bit unsigned Integer

A. 0 to 128
B. 0 to 127
C. 1 to 128
D. 1 to 127

Answer

B

Example 5

What is the range of n bit unsigned Integer

A. 0 to 2^{n-1}
B. 0 to 2^n - 1
C. 0 to 2^n-1 +1
D. -2^n +1 to 2^n-1

Answer

B

Example 6

What is the decimal value of unsigned 6-bit number 100000

A. 64

B. 32

C. 63

D. 31

Answer B

Example 7

What is the decimal value of unsigned 6 bit number 100011

A. 64

B. 32

C. 35

D. 31

Answer D

Example 8
What is the decimal value of unsigned 8-bit number 1000 0011

A. 131

B. 128

C. 259

D. 255

Signed Number

2'SCOMPLEMENT REPRESENTATION

(2)'s complement 2's complement = [(1's complement) + 1]

Binary Number	**1's complement**	**(2's complement)**
1011	1111-1011 = 0100	0100 + 1= 0101
0011101	1111111 – 0011101 = 1100010	1100010 + 1 = 1100011

Negative number are in 2's complement form

Example 1:

Suppose that n=8 and the binary representation 0 100 0001.

Sign bit is 0 ⇒ positive

Absolute value is 100 0001 = 65 Hence, the integer is +65

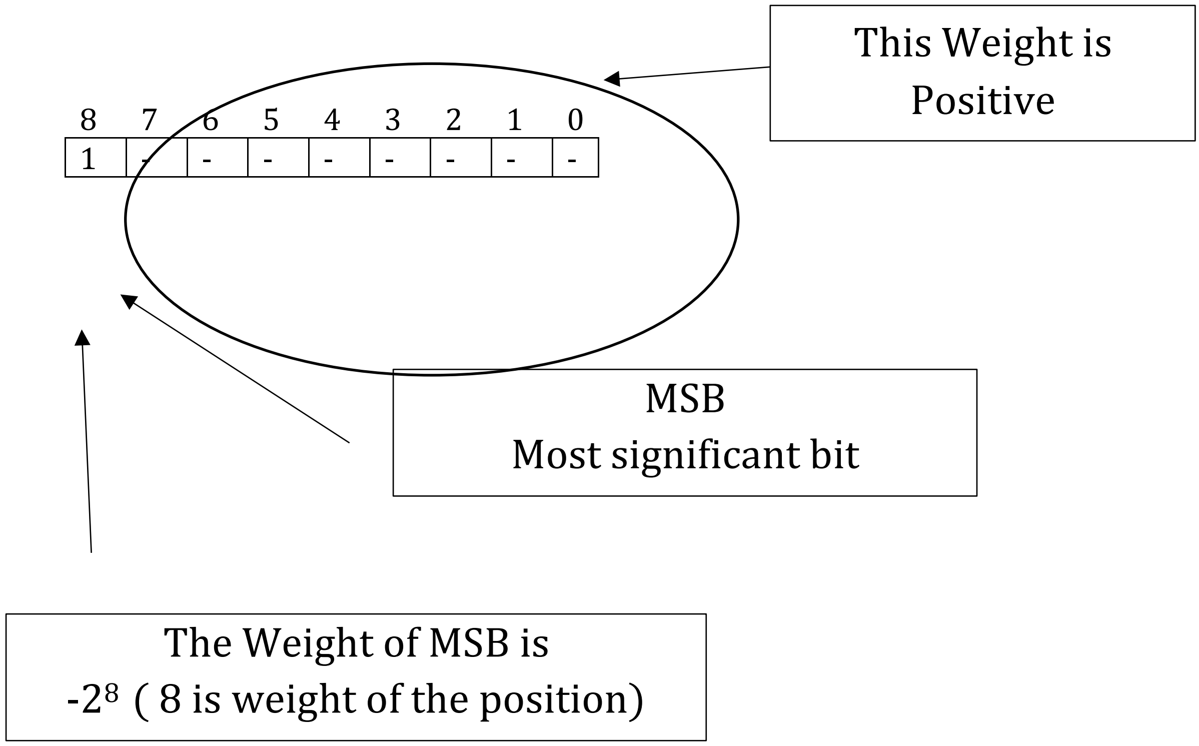
Example 2:

Suppose that n=8 and the binary representation 1 000 0001.

Sign bit is 1 ⇒ negative

Hence, the integer is -127

Weight of 1 in 2’s complement form



Example 1: Suppose that *n*=8 and the binary representation 0 100 0001.

Sign bit is 0 ⇒positive

Absolute value is 100 0001 = 65 Hence, the integer

is +65

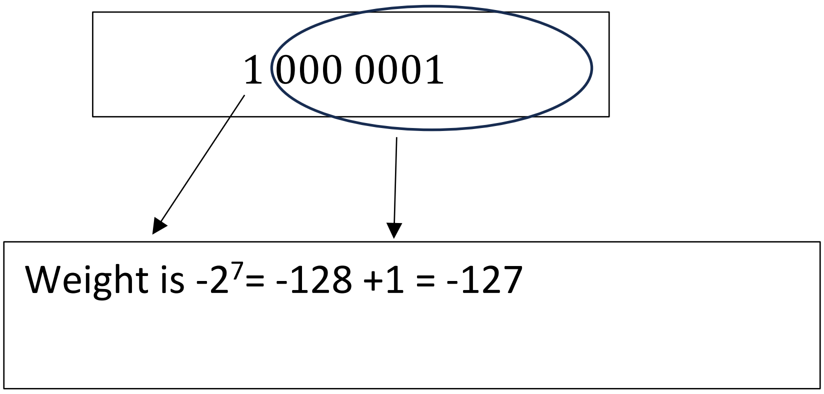
Example 2: Suppose that *n*=8 and the binary representation 1 000 0001.

Sign bit is 1 ⇒negative

Absolute value is the complement of 000 0001 plus 1, i.e., 111 1110 + 1 = 127

Hence, the integer is -127

nother way



1 000 0001

Weight is -2^7= -128 +1 = -127

Example 3

What is the decimal value of signed 6 bit number 100000

A. −64

B. −32

C. 64

D. 32

Answer
B

Example 4
What is the decimal value of signed 6 bit number 100011

A. 64

B. 29

C. -29

D. 31

Answer C

Example 5

What is the decimal value of signed 8 bit number 1000 0011

A. -125

B. -131

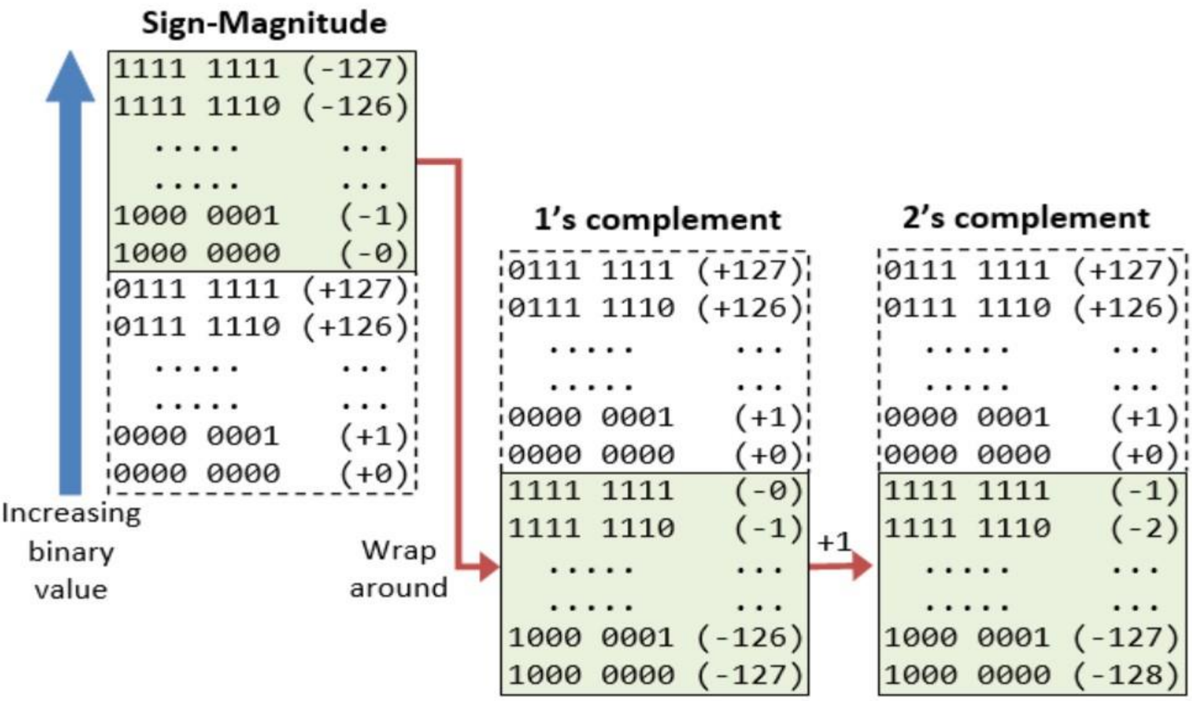
C. 259

D. -259

Answer

A

Signed Integer Representation

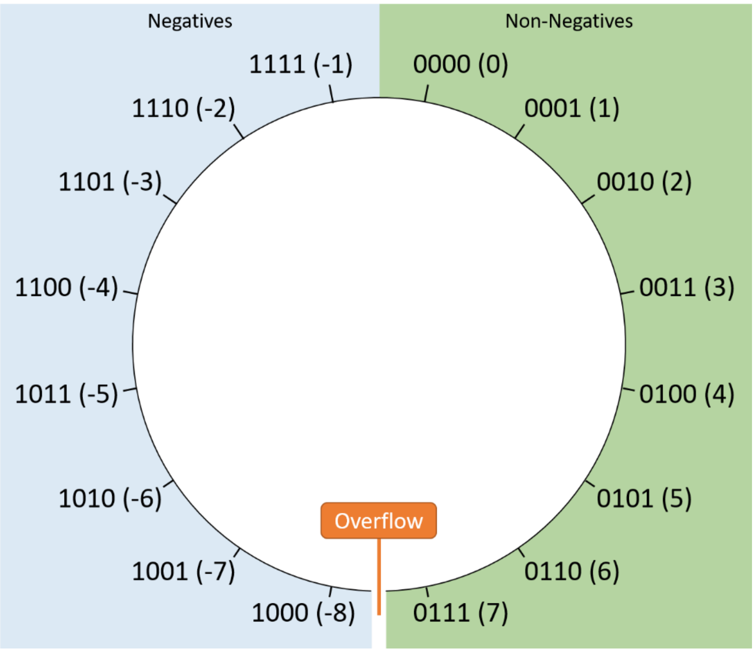


Range

An *n*-bit 2's complement signed integer can represent integers from

Range : $-(2^{n-1})$ to $+(2^{n-1} -1)$

n	minimum	maximum
8	-(2^7) (=-128)	+(2^7)-1 (=+127)
16	-(2^15) (=-32,768)	+(2^15)-1 (=+32,767)
32	-(2^31) (=-2,147,483,648)	+(2^31)-1 (=+2,147,483,647)(9+ digits)
64	-(2^63) (=-9,223,372,036,854,775,808)	+(2^63)-1 (=+9,223,372,036,854,775,807)(18+ digits) + digits)



If number is in range then correct output will be printed .

Example 1

What is the range of 8-bit signed Integer

A. 0 to 255

B. -128 to 127

C. -127 to 127

D. -126 to 128

Answer

B

-2^{n-1} to $2^{n-1}-1$ = -128 to 127

Example 2

What is the range of 16-bit signed Integer

A. 0 to 65536
B. -32768 to 32767
C. -32767 to 32767
D. -16384 to 16384

Answer

B

-2^{n-1} to $2^{n-1}-1$ = -32768 to -32767

Example 3

What is the range of n bit signed Integer

A. -2^n to 2^{n-1}

B. -2^{n-1} to $2^{n-1}-1$

C. $-2^n -1$ to 2^{n-1}

D. $-2^n +1$ to 2^{n-1}

Answer

B

Example 4

What is the decimal value of signed 6 bit number 100000

A. -64

B. -32

C. 64

D. 32

Answer

-32

Example 5

What is the decimal value of signed 6 bit number 100011

A. 64

B. 29

C. -29

D. 31

Answer

C

Explanation

First bit weight is negative rest of the bits weight is positive

-32+3 = 29

Example 6

What is the decimal value of signed 8-bit number 1000 0011

A. -125

B. -131

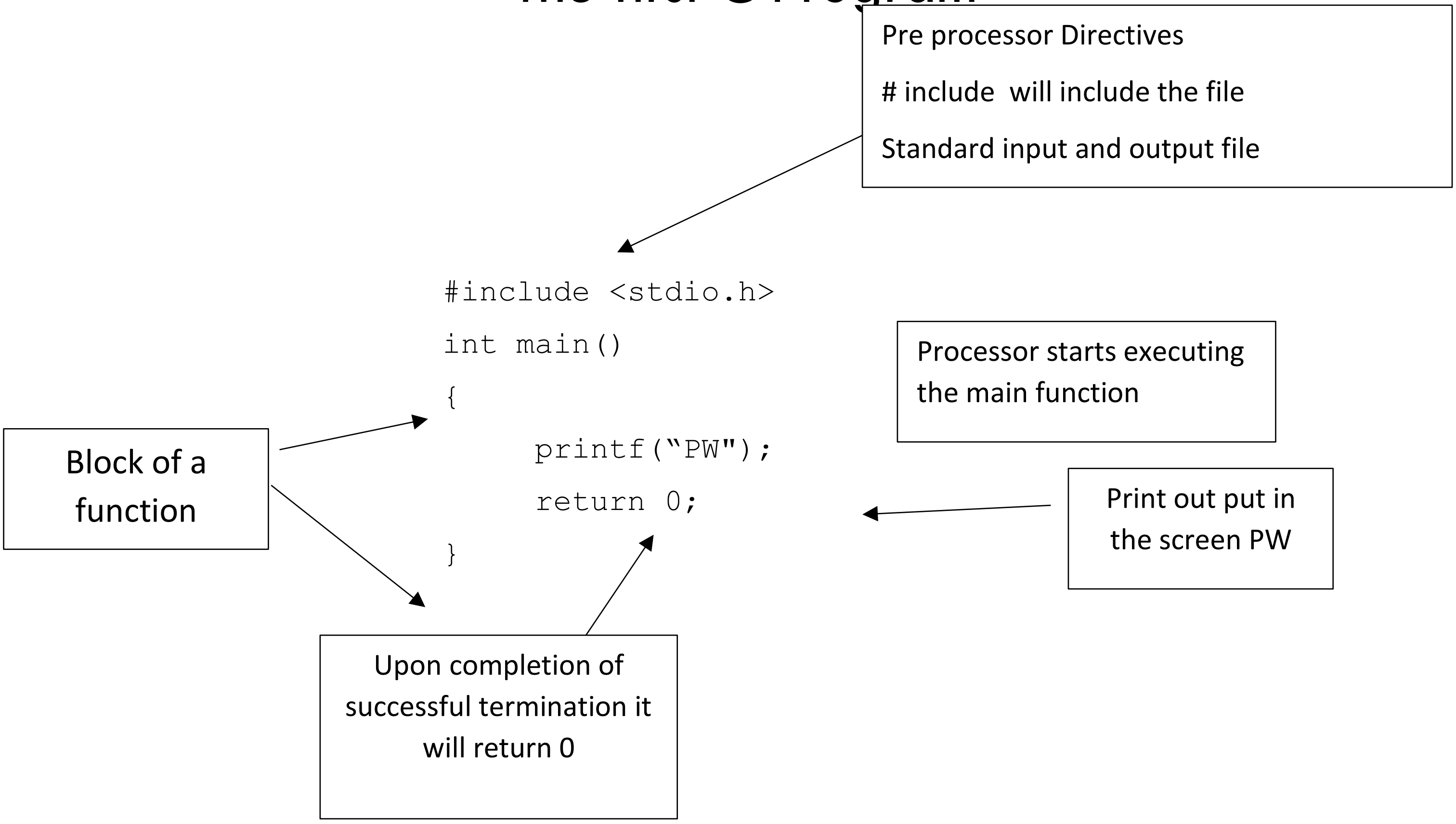
C. 259

D. -259

Answer

A

The first C Program



Pre processor Directives

# include will include the file

Standard input and output file

```
#include <stdio.h>
int main()
{
     printf(“PW");
     return 0;
}
```

Processor starts executing the main function

Block of a function

Print out put in the screen PW

Upon completion of successful termination it will return 0

Format specifies in C printf

A format specifier is a special sequence of characters that begins with a percentage sign (%) followed by a letter or a combination of letters. It indicates the data type and format of the variable passed as an argument in the printf() function.

Some common format specifiers in C language are:

- %c - for character data type
- %d or %i - for integer data type
- %f - for float data type
- %lf - for double data type
- %s - for string (array of characters)

- %hd – short int signed (2 Byte)
- %hu – short int unsigned (2 Byte)

Question 1

```
#include <stdio.h>
int main(){
        int a = 10;
        printf("%d", a);
        printf("%u", a);
        return 0;
}
```

This will print signed value of a

This will print unsigned value of a

Output:

Question 2

Short integer is of 2 Bytes

```
#include <stdio.h>
int main(){
        short int a = -10;
        printf("%hd", a);
        printf("%hu", a);
        return 0;
}
```

Output:

How To find over flow unsigned value in C language

Given number X

Suppose the given number is X

1. Calculate Remainder REM by dividing by 2^{16}
2. If the remainder is positive then that is the answer

3. If the remainder is negative then answer 65536 - |REM|

Question 3

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = -10;

        printf("%hu", a);

        return 0;

}
```

The output of the above program is -_____________

Answer

65536 – 10 = 65526

Question 4

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = -48;

        printf("%hu", a);

        return 0;

}
```

The output of the above program is -_____________

Answer

65536 – 48 = 65488

Question 5

Consider the following program

```
#include <stdio.h>
```

```
    int main(){
            short int a = 67000;
            printf("%hu", a);
            return 0;
    }
```

The output of the above program is -____________

Answer

The number is positive

67000 mod 2^{16} = 1464

Answer is in range

Example 6

Consider the following program

```
    #include <stdio.h>
    int main(){
            short int a = 140000;
            printf("%hu", a);
            return 0;
    }
```

The output of the above program is -____________

Answer

The number is positive

140000 mod 65536 = 8928

Answer is in range

Question 7

Consider the following program

```
    #include <stdio.h>
```

```
int main(){

        short int a = -70000;

        printf("%hu", a);

        return 0;

}
```

The output of the above program is -____________

Answer

The number is negative and not in the range

-70000 mod 65536 = -4464

Answer is

65536 – 4464=61072

Question 8

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = -88000;

        printf("%hu", a);

        return 0;

}
```

The output of the above program is -____________

Answer

The number is negative and not in the range

-88000 mod 65536 = -22464

Answer is

65536 – 22464=43072

How To find over flow signed value in C language

1. Given an integer x
2. Calculate the REMAINDER

   REM $= \text{X mod } 65536$

3. If the number is positive and in the range 0 to 32767

   Print the value REM

   Else number is negative

   $-(65336 - REM)$

4. If the remainder is negative and number in range $-1 \; to \; -32768$

   Then print the value

   Else number is positive

   And value is $= (65536 - |REM|)$

Question 1

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = 32771;

        printf("%hd", a);

        return 0;

}
```

The output of the above program is -_____________

Answer

32771 % 65536 = 32771

Number is positive but in not in the range

Hence answer is $= -(65536 - 32771) = -32765$

Question 2

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = 37780;

        printf("%hd", a);

        return 0;

}
```

The output of the above program is -_____________

Answer

37780 % 65536 = 37780

Number is positive but in not in the range

Hence answer is $= -(65536 - 37780) = -27756$

Question 3

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = -33333;

        printf("%hd", a);

        return 0;

}
```

The output of the above program is -_____________

Answer

-33333 % 65536 = -33333

Number is negative but in not in the range

Hence answer is $= (65536 - |33333|) = 32203$

Question 4

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = -64501;

        printf("%hd", a);

        return 0;

}
```

The output of the above program is -____________

Answer

-64501 % 65536 = -64501

Number is negative but in not in the range

Hence answer is $=(65536-64501)=1035$

Question 5

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = 167000;

        printf("%hd", a);

        return 0;

}
```

The output of the above program is -____________

Answer

167000 % 65536 = 35928

Number is positive but in not in the range

Hence answer is $=(65536-35928)=-29608$

Question 6

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = 98000;

        printf("%hd", a);

        return 0;

}
```

The output of the above program is -_____________

Answer

98000 % 65536 = 32464

Number is positive but in the range

Hence answer is = 32464

MSQ Multiple correct answer

Question   7

Consider the following program

```
#include <stdio.h>

int main(){

        short int a = 108000;

        printf("%hd", a);

        return 0;

}
```

Which of the following id TRUE for above program?

(A)The program will though overflow warning

(B)The program will print positive value

(C) The program will print negative value

(D)The out put will be -23071

Answer

(A),(C)

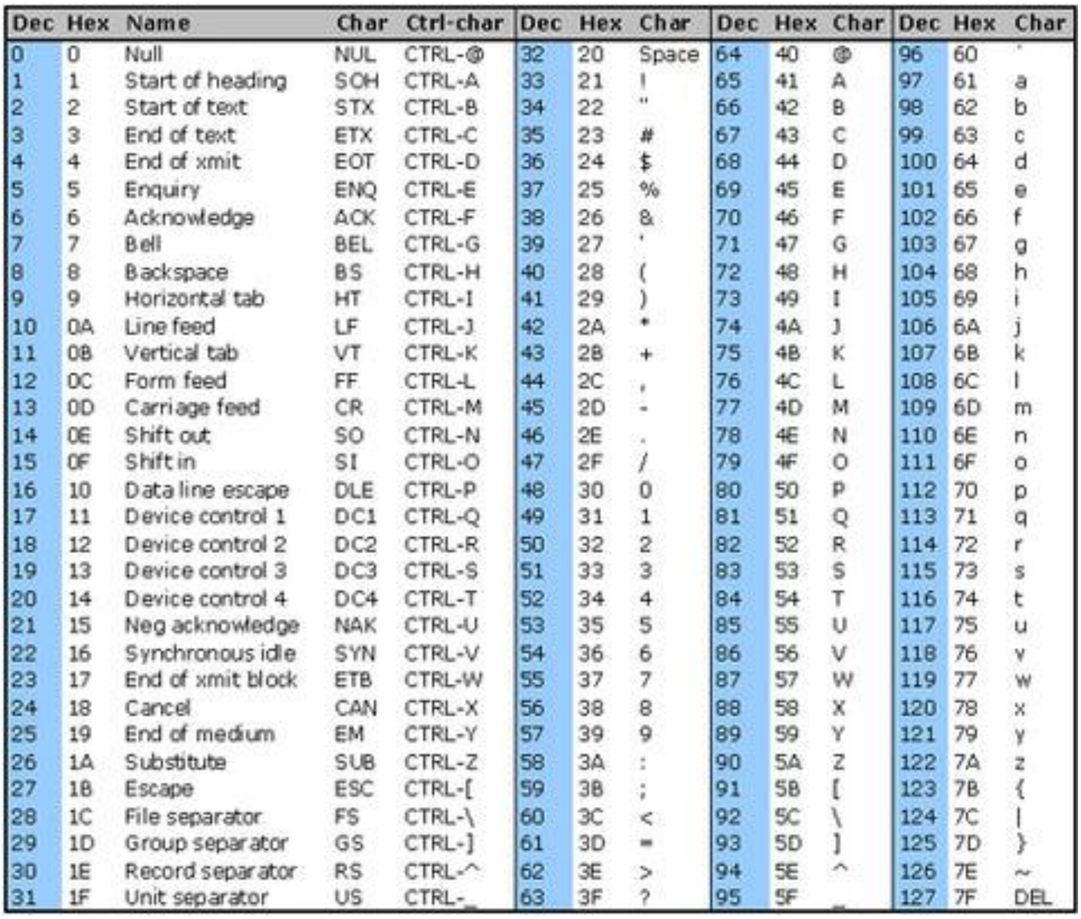
108000 % 65536 = 43464

Number is positive but not t in the range

Hence answer is $= -(65536 - 42464) = -23072$

Character in C Program

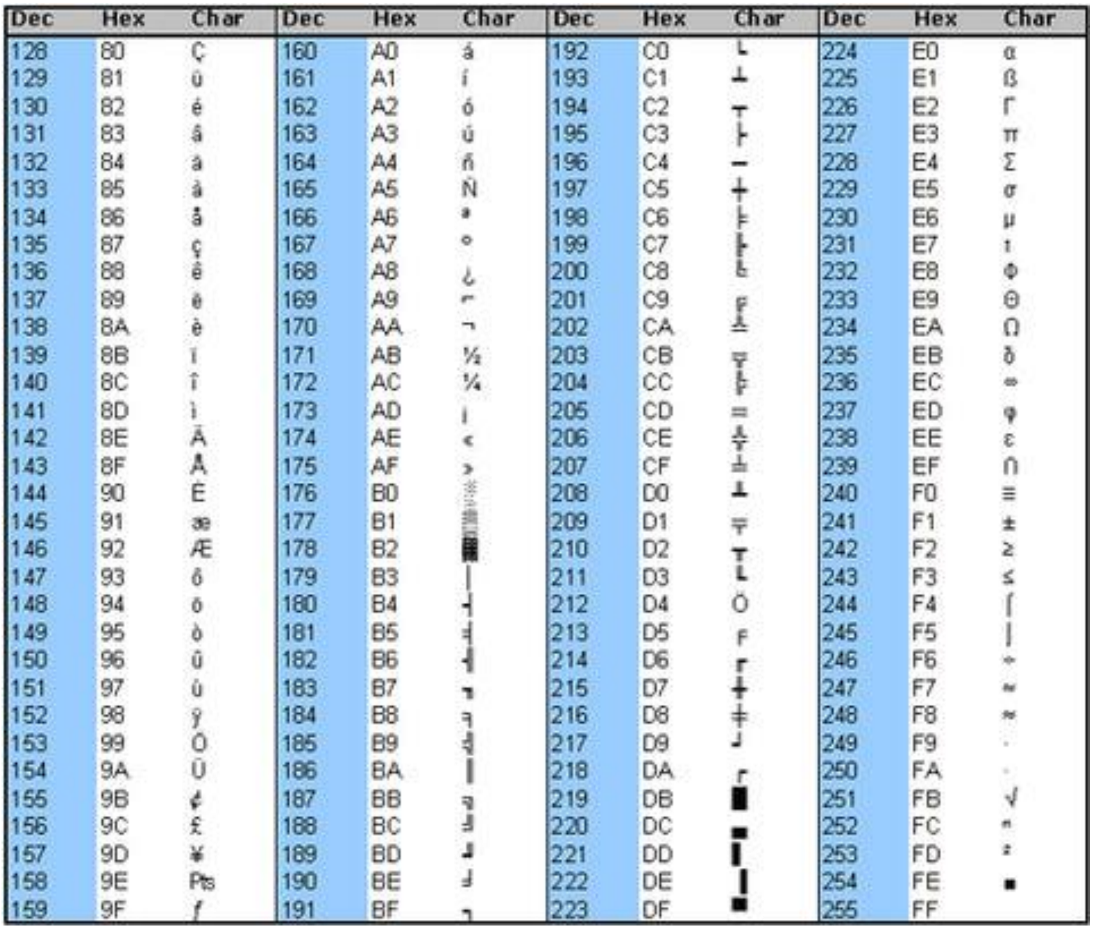
ASCII Value **ASCII**, a standard data-encoding format for electronic communication between computers. ASCII assigns standard numeric values to letters, numerals, punctuation marks, and other characters used in computers.



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Extended ASCII Table

Extended ASCII value is 8 bits



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	Ö	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	≈
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	·
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

The ADCII value of printable character is

0…9	48…57
A…Z	65…90
a…z	97…122

Example 1

```
#include <stdio.h>

#include <stdio.h>

int main(){

    char ch1 = 'a';

    char ch2 = 'z';

    printf("%c\n", ch1);

    printf("%d\n", ch1);

    printf("%c\n", ch2);

    printf("%d", ch2);

    return 0;
}
```

Output:

Example 2

```
#include <stdio.h>
int main(){
        char ch1 = ‘A';
        char ch2 = ‘Z';
        printf("%c\n", ch1);
        printf("%d\n", ch1);
        printf("%c\n", ch2);
        printf("%d", ch2);
return 0;
}
```

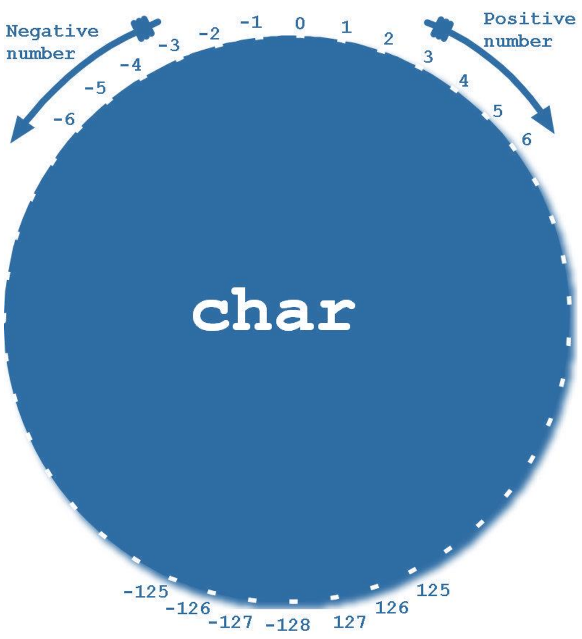
Output:

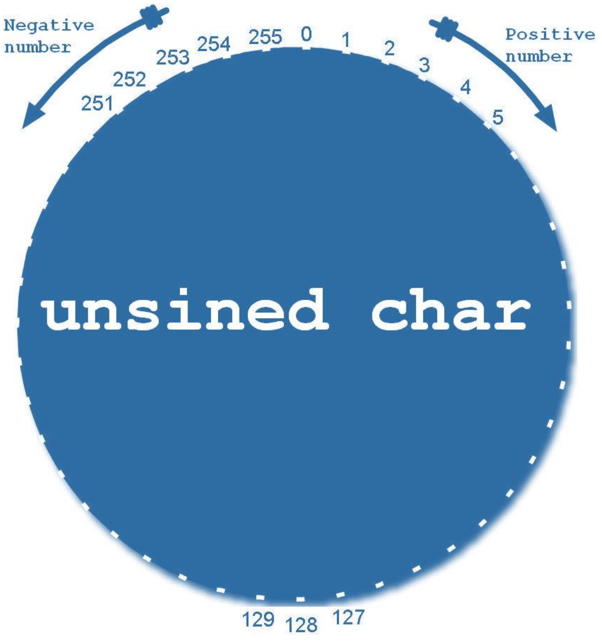
Example 3

```
#include <stdio.h>
#include <stdio.h>
int main(){
      char ch1 = ‘0';
      char ch2 = ‘9';
      printf("%c\n", ch1);
      printf("%d\n", ch1);
      printf("%c\n", ch2);
      printf("%d",
      return 0;
}
```

Output:

Range of character signed and unsigned





Unsigned and signed value represent the same thing

char a = 128;

char c = -128;

Print same value of character

char a = 129;

char c = -127;

Print same value of character

char a = 129;

char c = -127;

Print same value of character

MSQ

Question

If the char ch1 is assigned following values and character is printed then Which of the following is TRUE?

(A) 256 is 0 will print the same character

(B) 326 and 70 will print the same character

(C)139 and -117 will print the same character

(D) 145 and -112 will print the same character

Answer

(A) (B) (C)

Explanation:

Option A is 256 % 256 = 0 , second character is 0 only
Option b is 326 % 256 = 70
second character is 70 only both will print F

Option C 139 and unsigned value -117 is 256-117 = 139
Option D is Wrong

Question 2

The output of the program is

```
#include <stdio.h>
int main() {
char ch = –134;
printf(“%c”, ch);
return 0;
}
```

(A)a (B)x (C)y (D)z