Introduction to Programming I

Lab 3

Alexey Shikulin, Munir Makhmutov, Sami Sellami and Furqan Haider

Loops, Arrays and Pointers

Exercise 1:

What will be the output of the programs?

Case A

```
#include <stdio.h>
void swap(int *ap, int *bp) {
    int temp = *ap;
    *ap = *bp;
    *bp = temp;
}
int main() {
    int a = 1, *ap = &a;
    int b = 2, *bp = &b;

    swap(ap, bp);
    printf("%d %d %d %d\n", a, *ap, b, *bp);
    return 0;
}
```

Case B

```
#include <stdio.h>
void swap(int *ap, int *bp) {
    int *temp = ap;
    ap = bp;
    bp = temp;
}
int main() {
    int a = 1, *ap = &a;
    int b = 2, *bp = &b;

    swap(ap, bp);
    printf("%d %d %d %d\n", a, *ap, b, *bp);
    return 0;
}
```

Case C

```
#include <stdio.h>
int main() {
    int a = 1, *ap = &a;
    int b = 2, *bp = &b;

    int *temp = ap;
    ap = bp;
    bp = temp;

    printf("%d %d %d %d\n", a, *ap, b, *bp);
    return 0;
}
```

Exercise 1: Solution

What will be the output of the programs?

Case A

```
#include <stdio.h>
void swap(int *ap, int *bp) {
    int temp = *ap;
    *ap = *bp;
    *bp = temp;
}
int main() {
    int a = 1, *ap = &a;
    int b = 2, *bp = &b;

    swap(ap, bp);
    printf("%d %d %d %d\n", a, *ap, b, *bp);
    return 0;
}
```

2211

Case B

```
#include <stdio.h>
void swap(int *ap, int *bp) {
    int *temp = ap;
    ap = bp;
    bp = temp;
}
int main() {
    int a = 1, *ap = &a;
    int b = 2, *bp = &b;

    swap(ap, bp);
    printf("%d %d %d %d\n", a, *ap, b, *bp);
    return 0;
}
```

1122

Case C

```
#include <stdio.h>
int main() {
    int a = 1, *ap = &a;
    int b = 2, *bp = &b;

    int *temp = ap;
    ap = bp;
    bp = temp;

    printf("%d %d %d %d\n", a, *ap, b, *bp);
    return 0;
}
```

1221

Exercise 2:

What will be the output of the program?

```
#include <stdio.h>
int main() {
    int array[] = {10,20,30};
    int *pointer = array;

    printf("%d\n", *pointer);
    printf("%p\n", pointer);
    printf("%d\n", *array);
    printf("%p\n", array);

    printf("%d\n", ++*pointer);
    printf("%d\n", *++pointer);

    int *pointer1 = array;
    int *pointer2 = array;
    printf("%d\n", *pointer1++ + ++*++pointer2);
    return 0;
}
```

Exercise 2: Solution

What will be the output of the program?

```
#include <stdio.h>
int main() {
    int array[] = {10,20,30};
    int *pointer = array;

    printf("%d\n", *pointer);  // 10
    printf("%p\n", pointer);   // 0x7fff42
    printf("%d\n", *array);    // 10
    printf("%p\n", array);     // 0x7fff42

    printf("%d\n", ++*pointer);  // 11
    printf("%d\n", *++pointer);  // 20

    int *pointer1 = array;
    int *pointer2 = array;
    printf("%d\n", *pointer1++ + ++*++pointer2); // 32
    return 0;
}
```

Exercise 3:

Consider the following statements:

```
int *p;
int i;
int k;
i = 42;
k = i;
p = &i;
```

After these statements, which of the following statements will change the value of *i* to 75?

A) k = 75;
B) *k = 75;
C) p = 75;
D) *p = 75;

Exercise 3: Solution

Consider the following statements:

```
int *p;
int i;
int k;
i = 42;
k = i;
p = &i;
```

After these statements, which of the following statements will change the value of *i* to 75?

A) k = 75;
B) *k = 75;
C) p = 75;
D) *p = 75;

Exercise 4:

What will be the output of the program?

```
#include <stdio.h>
#include <string.h>
int main() {
    char buf1[100] = "Hello";
    char buf2[100] = "World";
    char *ptr1 = buf1+2;
    char *ptr2 = buf2+3;
    strcpy(ptr1, buf2);
    strcpy(ptr2, buf1);
    printf("%s\n", buf1);
    printf("%s\n", ptr1);
    printf("%s\n", buf2);
    printf("%s\n", ptr2);
    return 0;
}
```

Exercise 4: Solution

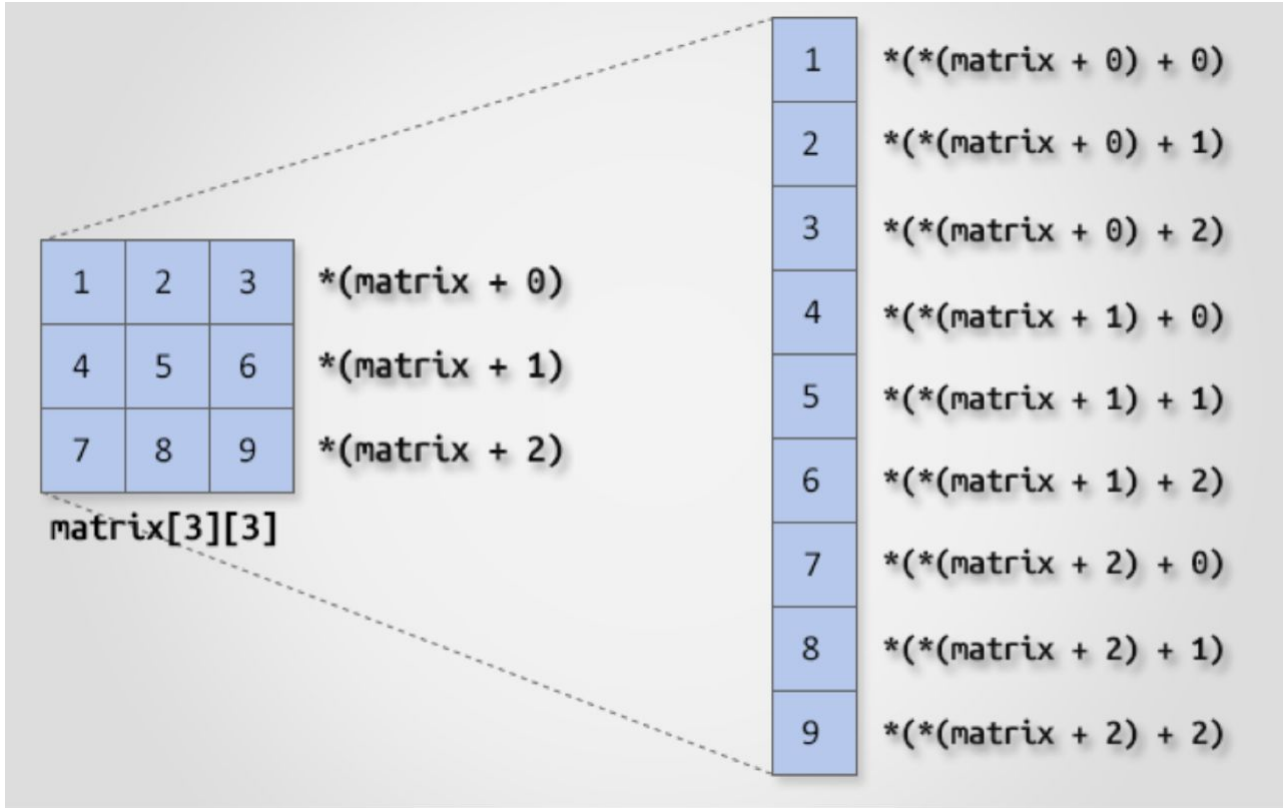
What will be the output of the program?

```
#include <stdio.h>
#include <string.h>
int main() {
    char buf1[100] = "Hello";
    char buf2[100] = "World";
    char *ptr1 = buf1+2;
    char *ptr2 = buf2+3;
    strcpy(ptr1, buf2);
    strcpy(ptr2, buf1);
    printf("%s\n", buf1); // HeWorld
    printf("%s\n", ptr1); // World
    printf("%s\n", buf2); // WorHeWorld
    printf("%s\n", ptr2); // HeWorld
    return 0;
}
```

Exercise 5:

Write a program to input and print elements of a two dimensional array using pointers and functions.

Exercise 5: Solution



Two dimensional array access using pointer

Exercise 5: Solution

```c
/**
 * C program to access two dimensional array using pointer.
 */

#include <stdio.h>

#define ROWS 3
#define COLS 3

/* Function declaration to input and print two dimensional array */
void inputMatrix(int matrix[][COLS], int rows, int cols);
void printMatrix(int matrix[][COLS], int rows, int cols);



int main()
{
    int matrix[ROWS][COLS];
    int i, j;

    /* Input elements in matrix */
    printf("Enter elements in %dx%d matrix.\n", ROWS, COLS);
    inputMatrix(matrix, ROWS, COLS);


    /* Print elements in matrix */
    printf("Elements of %dx%d matrix.\n", ROWS, COLS);
    printMatrix(matrix, ROWS, COLS);


    return 0;
}
```

```c
/**
 * Function to take input in two dimensional array (matrix)
 * from user.
 *
 * @matrix  2D array to store input.
 * @rows    Total rows in 2D matrix.
 * @cols    Total columns in 2D matrix.
 */
void inputMatrix(int matrix[][COLS], int rows, int cols)
{
    int i, j;


    for(i = 0; i < rows; i++)
    {
        for(j = 0; j < cols; j++)
        {
            // (*(matrix + i) + j is equivalent to &matrix[i][j]
            scanf("%d", (*(matrix + i) + j));
        }
    }
}



/**
 * Function to display elements of two dimensional array (matrix)
 * on console.
 *
 * @matrix  2D array to display as output.
 * @rows    Total rows in 2D matrix.
 * @cols    Total columns in 2D matrix.
 */
void printMatrix(int (*matrix)[COLS], int rows, int cols)
{
    int i, j;


    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
        {
            // *(*(matrix + i) + j) is equivalent to matrix[i][j]
            printf("%d ", *(*(matrix + i) + j));
        }

        printf("\n");
    }
}
```

Exercise 6:

Write a program to find the length of a string using pointer. Do not use strlen()

Exercise 6: Solution

Write a program to find the length of a string using pointer. Do not use strlen().

```
#include <stdio.h>

#define MAX_SIZE 100 // Maximum size of the string

int main()

{

    char text[MAX_SIZE]; /* Declares a string of size 100 */

    char * str = text; /* Declare pointer that points to text */

    int count = 0;

    printf("Enter any string: ");

    gets(text);

    /* Iterate through last element of the string */

    while(*(str++) != '\0') count++;

    printf("Length of '%s' = %d", text, count);

    return 0;

}
```

Exercise 7:

Write a program to copy one string to another using pointers. The program should stop when it finds an element of value 0. Do not use strcpy().

Exercise 7: Solution

```
#include <stdio.h>

#define MAX_SIZE 100 // Maximum size of the string

int main()

{

    char text1[MAX_SIZE], text2[MAX_SIZE];

    char * str1 = text1;

    char * str2 = text2;

    printf("Enter any string: ");

    gets(text1);

    /* Copy text1 to text2 character by character */

    while(*(str2++) = *(str1++));

    printf("First string = %s\n", text1);

    printf("Second string = %s\n", text2);

    return 0;

}
```

Exercise 8:

Write a program to print a histogram of the frequencies of different characters in its input. It is easy to draw the histogram with the bars horizontal; a vertical orientation is more challenging:

Input: hello world

Output:

```
h .
e .
l ...
o ..
w .
r .
d .
```

```
h e l o w r d
. . . . . . .
    . .
    .
```

Exercise 9:

Update previous solution, so that the output characters are sorted by frequencies in decreasing order. If the characters have the same frequency, sort by ASCII codes in increasing order.

Input: hello world

Output:

l ...
o ..
d .
e .
h .
r .
w .

Exercise 10:

Write a program which deletes duplicate elements from an array of integers.

Input: 0 1 1 3 1 1 4 2 5 4 0 0 5 **Output:** 0 1 3 4 2 5

Exercise 11:

Write a program to make such a pattern (triangle of height n, given as input) like a pyramid with numbers increased by 1

Input: 4 **Output:**

1
23
456
78910

Exercise 12:

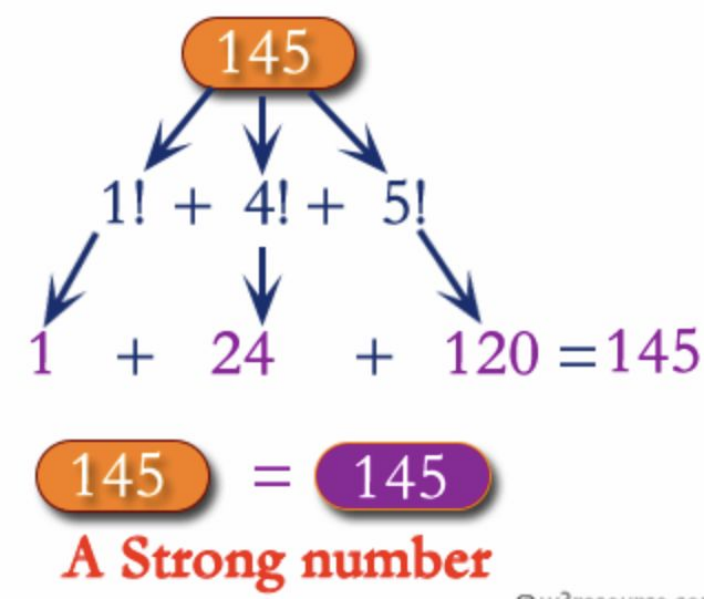
Write a program to find Strong numbers within a range of numbers.

Input:

Starting range: 1

Ending range: 200

Output: The strong numbers are: 1, 2, 145



If the sum of factorial of the digits in any number is equal to the given number, then the number is called a STRONG number.

Exercise 13:

Write a program that will try to find a user password using bruteforce. User password can be at least 1 symbol and at most 3 symbols and contains only ASCII characters from 32 to 126

Input:

password = u4!

Output:

found = u4!

number of attempts = …

ASCII (American Standard Code for Information Interchange)

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