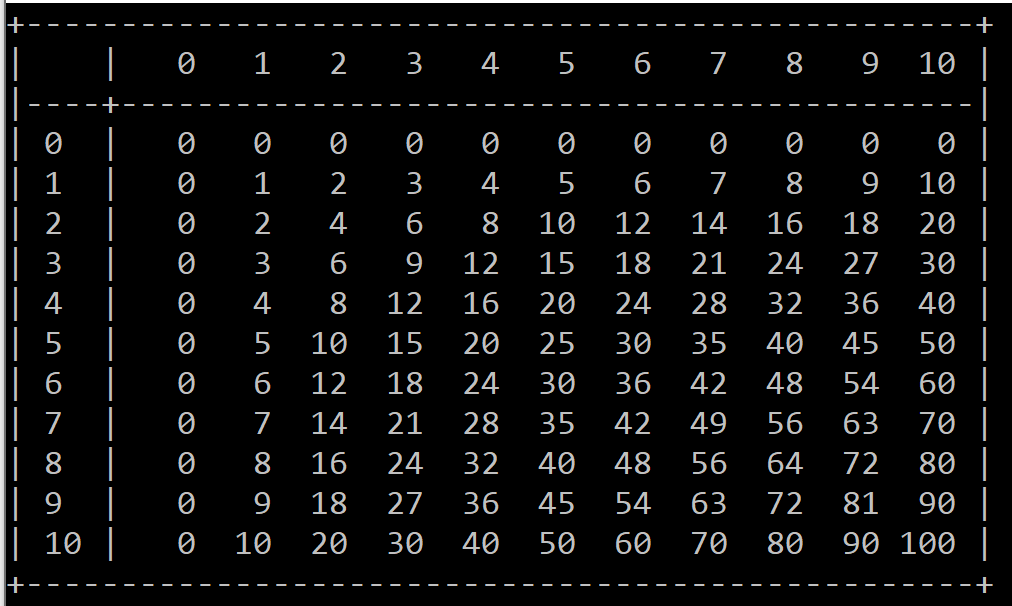
**מבחן 1 בשפת סיי**

**משך הבחינה: שעתיים  
משעה 10:55 עד 12:55  
יש לדאוג להגיש בזמן, אי הגשה בזמן יגרור הורדה בציון**

\*כתיבת הערות בשאלות 2-4 מקנה 2 נקודות בונוס לשאלה\*

**שם מלא:**

1. (15 נקודות) כתוב תוכנית אשר בונה את לוח הכפל הבא:



**(חובה להיות מדויקים בהדפסה! אי דיוק יגרור הורדה בציון)**

#include <stdio.h>

int main()

{

printf("+------------------------------------------------+\n");

printf("| | 1 2 3 4 5 6 7 8 9 10 |\n");

printf("|------------------------------------------------|\n");

printf("| 0 | 0 0 0 0 0 0 0 0 0 0 |\n");

for (int i = 1; i <= 10; i++)

{

printf("| %-4d|", i);

for (int n = 1; n <= 10; n++)

{

printf("%4d", i \* n);

}

printf(" |\n");

}

printf("+------------------------------------------------+\n");

return 0;

}

2. (20 נקודות) כתוב תוכנית הקולטת מהמשתמש מספר חיובי שלם N ולאחר מכן קולטת ממנו N תווים מופרדים ע"י ENTER. על התוכנית להתעלם מכל תו שהוא לא אות באנגלית ובסוף להדפיס על המסך את האות הממוצעת.  
לדוגמא עבור הקלט הבא:

a  
b  
C  
d  
E  
3  
^  
1

output: c

**הבהרות**:   
לצורך קליטה יש להשתמש ב:

scanf(" %c", &ch);

כאשר אנחנו עושים קליטת תווים עם הפרדה של ENTER יש לעשות **רווח** לפני ה-%, דבר זה מסמן התעלמות מכל סימני רווח, TAB, ENTER קודמים שהוכנסו (וממתינים בתוך ה stdin לקליטה), אשר עלולים להזיק בזמן קליטת תווים **בלולאה**.

יש לחשב ממוצע עבור המיקומים של האותיות באלף בית האנגלי בלי קשר לצורה של האות (capital/small). יש לדאוג לעגל את המספר למעלה או למטה בהתאם לקרבה (הרי לא יכול להיות אות במיקום ה 3.7 באלף בית, לכן נעגל כלפי מעלה לקבלת המספר 4 אשר יוצר את האות e).  
צורת ההדפסה של האות הממוצעת (capital/small) מוחלט על פי איזה צורה הופיע יותר, אם היו יותר אותיות קטנות מגדולות (או שיוון) אז נדפיס את האות הממוצעת גם כן כאות קטנה, אחרת נדפיס בצורה הגדולה (capital)

3. (30 נקודות)  
א. (10 נקודות) כתוב פונקציה שמקבלת מספר שלם num ומספר שלם digit (בגודל 1-9), על הפונקציה להחזיר 1 במידה והספר digit נמצא בדיוק digit פעמים בתוך num. אחרת תחזיר 0. לדוגמא עבור num שהוא 3233592 ועבור digit 2 או 3 אז הפונקציה תחזיר 1, ועבור 9 היא תחזיר 0.  
ב. (20 נקודות) כתוב פעולה ראשית שתקלוט מספר שלם וחיובי N ומספר שלם digit (מיצג ספרה מ1 עד 9). ולאחר מכן תקלוט N מספרים שלמים וחיוביים, ותציג כפלט את מספר המספרים שהספרה digit מופיעה בהם בדיוק digit פעמים.  
**אין להניח תקינות קלט**

#include <stdio.h>

int check\_input(int num);

int check\_digit(int num, int digit);

int main()

{

int num, digit, input\_res = 0, counter = 0, finish = 0;

while (input\_res < 1)

{

printf("Enter number <0+> : ");

input\_res = scanf("%d", &num);

if (!input\_res || num <= 0)

{

printf("Error - invalid input..\n");

input\_res = 0;

}

}

input\_res = 0;

while (!finish)

{

printf("Enter digit <1-9> (-1 to break): ");

input\_res = scanf("%d", &digit);

if (digit == -1)

{

printf("The sum of current digits : %d", counter);

finish = 1;

continue;

}

else

{

if (input\_res < 1 || !check\_input(digit))

{

printf("Error - invalid input..\n");

input\_res = 0;

continue;

}

else if (check\_digit(num, digit))

{

counter++;

}

}

}

return 0;

}

int check\_input(int digit)

{

int input\_res = (digit >= 1 && digit <= 9) ? 1 : 0;

return input\_res;

}

int check\_digit(int num, int digit)

{

int counter = 0, res;

while (num)

{

int n = num % 10;

if (n == digit) counter++;

num = num / 10;

}

res = counter == digit ? 1 : 0;

return res;

}

4. (35 נקודות) מספר טבעי (שלם חיובי) ייקרא "מספר עוקב עולה" אם הוא מורכב מספרות עוקבות. כלומר כל ספרה גדולה ב 1 מקודמתה. למשל, המספרים 1234, 345678, 6 הם מספרים עוקבים עולים. והמספרים 68, 1278, 9876 אינם מספרים עוקבים עולים.   
אותו עקרון עבור "מספר עוקב יורד", המספרים 4321, 7654, 4 הם מספרים עוקבים יורדים, ו 4312, 3456 הם לא.  
א. (25 נקודות) כתבו פעולה שתקבל מספר טבעי. הפעולה תחזיר את הערך UP (יש להגדירו כקבוע) אם המספר הוא "מספר עוקב עולה", ותחזיר DOWN (קבוע) אם המספר הוא "מספר עוקב יורד", ותחזיר BOTH (קבוע) אם המספר הוא גם עולה וגם יורד, ותחזיר FALSE (כקבוע) אם המספר איננו "מספר עוקב".

#include <stdio.h>

#define UP 2

#define DOWN 1

#define BOTH 0

#define FALSE -1

int check\_sequence(int num);

int main()

{

int num, input\_res = 0;

while (input\_res < 1)

{

printf("Enter number <int> : ");

input\_res = scanf("%d", &num);

if (!input\_res)

{

printf("Error - invalid input..\n");

input\_res = 0;

}

}

printf("%d", check\_sequence(num));

return 0;

}

int check\_sequence(int num)

{

int n1, res = -2, seq;

if (num / 10 == 0) return BOTH;

n1 = num % 10;

num /= 10;

while (num)

{

int n = num % 10, seq;

switch (n - n1)

{

case 1:

seq = UP;

break;

case -1:

seq = DOWN;

break;

default:

seq = -1;

break;

}

if (res == -2 || res == seq)

{

res = seq;

}

num /= 10;

}

return res;

}

ב. (10 נקודות) כתוב תוכנית אשר תקלוט מהמשתמש מספר שלם N (שלילי או חיובי), ולאחר מכן תקלוט N מספרים שלמים (שליליים וחיוביים) ולאחר **כל** קליטה תדפיס:

This number is a Up-Sequence number

**או**

This number is a Down-Sequence number

**או**

This number is both Up and Down-Sequence number

**או**

NOT a sequence number!

בהתאם לערך המוחזר. **על התוכנית לעבוד גם עם קליטות חיוביות וגם שליליות**

#include <stdio.h>

#define UP 2

#define DOWN 1

#define BOTH 0

#define FALSE -1

int check\_sequence(int num);

int main()

{

int num, num2, input\_res = 0, finish = 0;

while (input\_res < 1)

{

printf("Enter number <int> : ");

input\_res = scanf("%d", &num);

if (!input\_res)

{

printf("Error - invalid input..\n");

input\_res = 0;

}

}

while (1)

{

printf("Enter the next number <int> : ");

input\_res = scanf("%d", &num2);

if (!input\_res)

{

printf("Error - invalid input..\n");

input\_res = 0;

}

switch (check\_sequence(num, num2))

{

case -1:

printf("NOT a sequence number!\n");

break;

case 0:

printf("This number is both Up and Down-Sequence number\n");

break;

case 1:

printf("This number is a Up-Sequence number\n");

break;

case 2:

printf("This number is a Down-Sequence number\n");

break;

}

num = num2;

}

return 0;

}

int check\_sequence(int num, int num2)

{

int res;

switch (num - num2)

{

case 0:

res = BOTH;

break;

case 1:

res = UP;

break;

case -1:

res = DOWN;

break;

default:

res = -1;

break;

}

return res;

}