תרגיל

דקודים:

```
void f(int i)
{ if (i==0) return;
  cout<<i--;
  f(i);
}
```

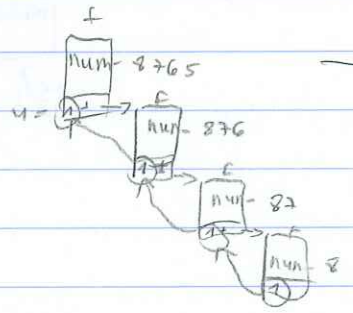
רקורסיה - פעולה חוזרת

כדי לקבל פונקציה ועוד פונקציה,

אם יש פעולה שחוזרת עד יותר קל לעשות רקורסיה

```
int f(int num)
{ if (num<10)
    return 1;
  return 1+f(num/10);
```

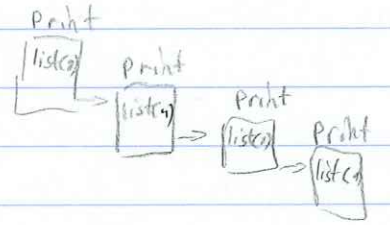
num = 8765



מחזירה כמה

ספרות יש בה

```
void Print (item *list)
{ if (list==0)
    return;
  print(list->next)
  cout<<list->x;
}
```




```
int func(int num)
{ if (num<10)
    return num;
  int x=num%10;
  int y=fun1(num/10);
  if (x>y)
    return x;
  return y;
```

סוף

המשך-

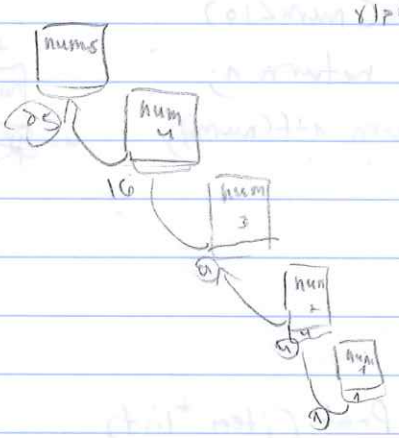
```
int funct(int num)
{ if (num==1)
    return 1;
  return funct(num-1) + 2^(num-1);
```

num=5

מקרה

יסוד איטרציה

מקרה קיצון



בס"ד

רקורסיה

רקורסיה היא תהליך שבו פונקציה קוראת לעצמה במילים אחרות פונקציה הכוללת הוראת קריאה לעצמה נקראת פונקציה רקורסיבית.

פונקציה רקורסיבית כוללת הוראת קריאה לעצמה הפונקציה ותנאי לסיום הרקורסיה , שימי לב חייב להיות תנאי עצירה אחרת תוצר לולאה אין סופית יש לדאוג שתנאי העצירה יתממש בשלב כלשהוא .

מה קורה במחשב בעת הפעלת פונקציה מפונקציה ? לדוגמא f1 מפעילה את f2

- המחשב שומר לעצמו כתובת חזרה של הפקודה הבאה בf1 אותה הוא צריך לעשות לאחר סיום f2
- המחשב משאיר את המשתנים הפנימים של f1 שכן היא עדיין לא הסתימה
- לפונקציה f2 מוקצה מקום חדש בזיכרון למשתנים שלה.

בעת הפעלת פונקציה רקורסיבית מתבצע אותו תהליך

- המחשב שומר לעצמו כתובת חזרה של הפקודה הבאה אותה הוא צריך לעשות לאחר סיום הפעלת הגרסה החדשה.
- המחשב משאיר את המשתנים הפנימים של הגרסה הנוכחית שכן עדיין היא לא הסתימה
- לפונקציה החדשה מוקצה מקום חדש בזיכרון למשתנים הפנימים שלה.

הפקודות הרשומות בפונקציה עד שלהפעלת הרקורסיה-הפעלת הפונקציה את עצמה יופעלו שוב ושוב עד לעומק הרקורסיה , כלומר עד שתנאי העצירה יתממש ולאחר מכן יתבצעו הפקודות שנותרו בסדר הפוך מעומק הרקורסיה עד לגרסה הראשונה.

פונקציות רקורסיביות יעילות לפתרון בעיות המכילות אופי רקורסיבי. ז"א בעיות שחוזרות ע"ע בדרגת קושי גדלה והולכת ובעיות בהן כל צעד מסתמך על הצעד שלפניו.

<u>תכנון פתרון רקורסיבי:</u>

1. הניחי שיש פונקציה שפותרת את הבעיה לדרגת הקושי הקודמת
2. השתמשי בה על מנת לפתור את בעית הקושי הנוכחית
3. הוסיפי תנאי עצירה , כלומר מה יש לעשות כאשר אין דרגת קושי קודמת
4. הפונקציה שרשמת ב3 השלבים הקודמים היא הפונקציה הרקורסיבית

<u>תרגול</u>

לפניכן כמה פעולות רקורסיביות עבור כל אחת מהן רשמי את טענת היציאה

```
void func1(char ch, int n)
{
      if (n > 0)
      {
            cout << ch;
            func1(ch, n - 1);
      }
}
```

טענת כניסה הפעולה מקבלת תו ומספר חיובי שלם

טענת יציאה הדפסה ח פעמים בן מספר

טענת כניסה הפעולה מקבלת מספר חיובי שלם

טענת יציאה הדפסת מקומות

```
int func2(int n)
{
        if (n < 10)
                return n;
        int x = n % 10;
        int y = func2(n / 10);
        if (x > y)
                return x;
        else return y;
}
```

טענת כניסה הפעולה מקבלת מספר חיובי שלם
טענת יציאה מחזיר את המקסימום

```
int func3(int n)
{
        if (n == 1)
                return 1;
        return (func3(n - 1)) + 2 * n - 1;
}
```

טענת כניסה הפעולה מקבלת מספר חיובי שלם
טענת יציאה מחזיר את המספר הגדול ביותר (שלם)

```
int func4(int n)
{
        if (n < 10)
                return n;
        int i = 10;
        while (n%i != n)
                i *= 10;
        return ((n % 10)*i / 10) + func4(n / 10);
}
```

השלימי את השורות הריקות בפונקציה על פי טענת היציאה.
טענת כניסה-הפונקציה מקבלת מספר שלם חיובי
טענת יציאה- הפונקציה מחזירה את סכום הספרות של המספר המתקבל

```
int func5(int n)
{
        if (n < 10)
                return n;

        return (n % 10)+ func5(n/10);
}
```

טענת כניסה-הפונקציה מקבלת מספר שלם חיובי
טענת יציאה- הפונקציה מחזירה את מספר הספרות של המספר המתקבל

```
int func6(int n)
{
        if (n < 10)
                return 1;

        return 1 + func6(n/10);
}
```

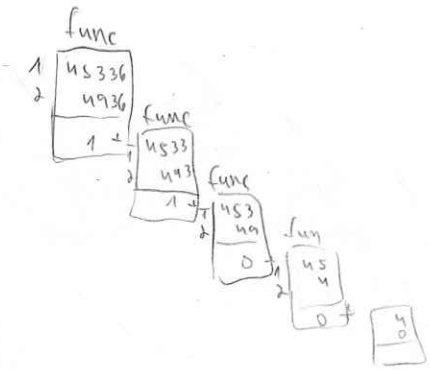
בס"ד

ציני נכון /לא נכון למשפטים הבאים:

1. כאשר מסתימת פונקציה, התוכנית חוזרת לפונקציה השולחת לפקודה הבאה. נכון
2. לכל פונקציה שמופעלת מוקצה מקום בזיכרון לשמירת ערכי המשתנים הפנימים והפרמטרים שלה, מקום זה משתחרר רק בסיומה. נכון
3. יכול להיות למספר פונקציות משתנה פנימי באותו שם, אין קשר בין המשתנים הללו. נכון
4. כאשר מסתימת פונקציה כל שהיא כל המשתנים הפנימים בה נעשים בלתי מוכרים. לא נכון
5. כאשר מסתימת פונקציה כל שהיא כל המשתנים הפנימים בה אובדים . נכון
6. כאשר התוכנית עוברת לביצוע פונקציה כלשהיא כל המשתנים הפנימים בפונקציה הקודמת נעשים בלתי מוכרים. נכון
7. כאשר התוכנית עוברת לביצוע פונקציה כלשהיא כל המשתנים הפנימים בפונקציה הקודמת אובדים. לא נכון
8. כאשר התוכנית עוברת לביצוע פונקציה כלשהיא מסתימת הפונקציה הקודמת וכל המשתנים הפנימים בפונקציה הקודמת אובדים. לא נכון
9. הערכים במשתנים הפנימים של כל פונקציה נשמרים עד לסיומה כלל זה תקף גם אם פונקציה זו מפעילה פונקציה אחרת. נכון
10. בזמן ריצה יתכן ויהיו מספר פונקציות שהן באמצע הביצוע נכון

שאלות

1. רשמי פונקציה המקבלת 2 מספרים שלמים חיובים ומחזירה את מספר הספרות השוות בערכן ובמקומן ב2 המספרים
2. פונקציה המקבלת מספר שלם ובודקת האם כל הספרות במספר זוגיות
3. פונקציה המקבלת 2 מספרים שלמים חיובים ומחזירה את תוצאת החלוקה בשלמים.
4. פונקציה המקבלת מספר שלם ומדפיסה אותו בבינארית.



בס"ד

מה מבצעת הפונקציה
מה ההבדל בין הפונקציה הראשונה לשניה
מה יקרה בעת הפעלת הפונקציה השלישית מה היא מחליפה?

```
void f(int x, int y)
{cout << x << " ";
    if (x <= y)
        f(x + 1, y);

}

void f_1(int x, int y)
{

    if (x <= y)
        f_1(x + 1, y);
    cout << x << " ";

}
void f1(int a)
{

    if (a < 0)
        return;

    f(1, a);
    cout << "\n";
    f1(a - 1);

}
int main()
{

    for (int i = 0; i < 9; i++)
    {
        f(1, i);
        cout << "\n";
    }cout << "\n";

f1(9);
    return 0;

}
```

חלק ד'

45 נקודות

ענה על 3 מתוך השאלות 5-8

שאלה 5

```
int what (int num, int digit)
{
        if (num <10)
                if (num ==digit)
                        return 1;
                else
                        return 0;
        else
                return what(num /10,digit);
}

void display(int a[],int low,int high)
{
if (low <= high)
        {
                if (what(a[low],a[low]%10)==1)
                        printf("%d  ",a[low]);
                display (a,low+1, high);
        }
        else
                printf("***");
}

void main()
{
        int a[]={212,123,2223,4,98,4554,546,22,19,5};
        display(a,0,9);
}
```

א. מה מבצעת הפונקציה what?

ב. מה פלט התוכנית, כתבי מעקב אחר ביצוע הפונקציה display.

ג. מה מבצעת התוכנית?

שאלה 3

כתבי פונקציה לניהול לוח פגישות

הפגישות לפי סדר שעות היום ביום מסוים. הפגישות מתחילות החל מ – 9:00 בבוקר בשעון של 24 שעות.

הפונקציה מקבלת רשימה מקושרת חד כיוונית הכוללת בקשות לפגישה לפי סדר קבלת הבקשות (לא לפי שעות). המידע על כל פגישה כולל: שעת התחלה (שעה עגולה) ומשך הפגישה בשעות. יש לבנות רשימה חדשה שתהווה את לוח הפגישות. הרשימה תכיל את פרטי הפגישות שניתן לבצע, ממוינות עפ"י שעות היממה.

כאשר קיימת התנגשות בין בקשות ישנה עדיפות לפגישה שבקשתה התקבלה קודם. פגישה שהועברה ללוח הפגישות נמחקת מהרשימה. ברשימה תשארנה הפגישות שנדחו.

על הפונקציה להדפיס את לוח הפגישות שנקבעו לאותו יום, לכל פגישה שעת התחלה ומשך הפגישה. וכן מספר הפגישות שנדחו.

שאלה 3

במכשיר הטלפון שמורה <u>רשימת אנשי קשר</u> (ממוינת מילונית).

```
struct person
{
        char name[20];
        person *next;
};
```

במכשיר קיימים מקשי מספרים (0-9) וכל מספר מסמל אפשרות של אחת מתוך קבוצת האותיות הרשומה עליו.
הייצוג: מערך של מחרוזות char keys[10][5]. מיקום במערך מייצג את המספר המתאים. לדו' keys[4]="GHI"

איתור איש קשר ע"י המשתמש מתבצע ע"י הקשת מספר ללא בחירת אות מסוימת. בכל הקשת מספר המכשיר מציג מתוך רשימת אנשי הקשר את השמות המתאימים לכל האותיות שבמספר. עם כל הקשת מספר מצטמצמת רשימת השמות האפשריים.

כתבי פונקציה שכותרתה: void ContactList (person *plist , char keys[ ][5])
הפונקציה מאפשרת קלט מספרים מהמשתמש. לאחר הקשת מספר יוצרת את רשימת השמות המתאימים לבחירה.
ברשימה זו כל פריט מכיל **<u>מצביע</u>** לפריט מרשימת אנשי הקשר. בכל שלב הפונקציה מדפיסה את רשימת השמות.
יש להסתמך על הרשימה מהשלב הקודם ולמחוק ממנה את הפריטים שאינם מתאימים להקשה החדשה.
הקלט מסתיים עם הקשת '#' או כשרשימת האפשרויות ריקה.

הנחה: כל הנתונים באותיות גדולות בלבד

דוגמא רשימת אנשי הקשר:

AKUKA, ALFA, ALONI, AVNI, BERGER, BLOY, CKOCK, DADON, FRID, FAIG, GOLD , HOFT, MEIR

עבור הקלט: 2,5,6,2,3

כשיוקש המספר 2 תוצג הרשימה: <u>A</u>KUKA, <u>A</u>LFA , <u>A</u>LONI, <u>A</u>VNI ,<u>B</u>ERGER, <u>B</u>LOY, <u>C</u>KOCK

כשיוקש המספר 5 תוצג הרשימה: A<u>K</u>UKA, A<u>L</u>FA, A<u>L</u>ONI, B<u>L</u>OY, C<u>K</u>OCK

כשיוקש המספר 6 תוצג הרשימה: AL<u>O</u>NI, BL<u>O</u>Y, CK<u>O</u>CK

כשיוקש המספר 2 תוצג הרשימה: CKO<u>C</u>K ,

כשיוקש המספר 3 תיווצר רשימה ריקה ויסתים החיפוש.

שאלה 8

1. מה מבצעת הפונקציה הבאה?
2. כתבי 2 קריאות לפונקציה :
   א. עבורה תחזיר 0 אין כזאת
   ב. עבורה תחזיר 3 שהוא מופיע כ. פעמים

```
int rstr(char* st,const char*s)
{
        char* ptr=strstr(st,s);
        if (ptr==NULL)
                return 0;
        strcpy(ptr,ptr+strlen(s));
        return (rstr(st,s)+1);
}
```

לא זה כתוב שזה שלם
מוחק מקבל פקודות מערכת.

char *ptr
strcpy(ptr)

ptr"

abcadg
adg

adg
abcadg

abc

ptr=

תרגיל

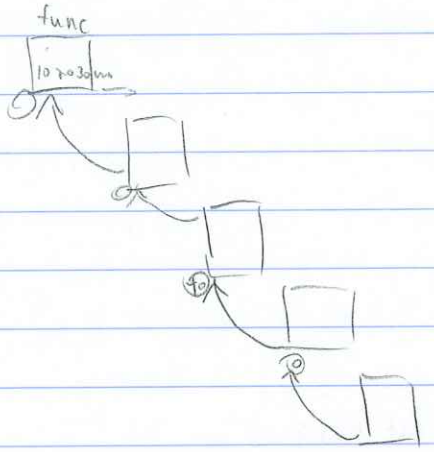
כתבי בונקציה

שקבלת מערך ומחזירה סכום איברים קיימים בלי שינוי בלולאה.

```
int func (int *arr, int lengh)
{
    if(lengh==0)
        return 0;
return (*arr + func(arr+1, lengh-1))
```

lengh = 4

arr 10 20 30 40



סוף

רשת:

פונקציה שמקבלת מחרוזת ואורך ומחזירה האם פלינדרום או לא

```
int func(char *str, int len)
{
    if(len<2)
        return 1;
    return *str==*(str+len-1) && func(str+1, len-2);
}
```

abba

func
str-abba
len-4

func
bba
2

func
ba
0

func

- void func(int num)

```
{
    if(num==0)
        return;
    func(num/2)
    cout<<num%2
}
```

num=6

6

3

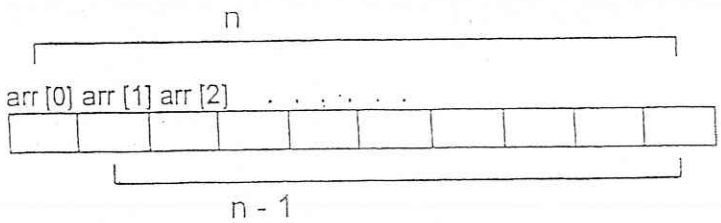
1

0

מתקדם

רקורסיה

נסתכל על מערך כנ"ל ,נאמר arr :

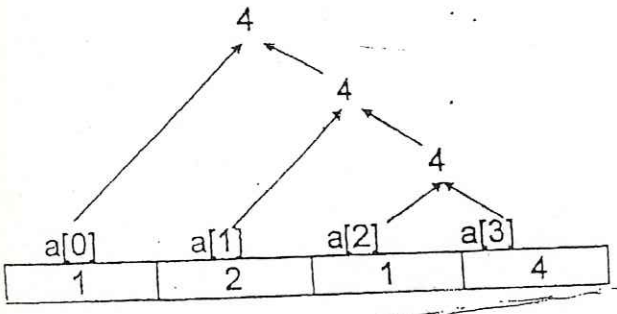


מימד הבעיה במקרה זה הוא גודל המערך, כלומר n. ניתן לראות שאם נתון המקסימום במערך שגודלו n-1 ושמתחיל מ – arr[1] אזי המקסימום בין ערך זה לבין arr[0] הוא המקסימום במערך שגודלו n ומתחיל ב – arr[0]. כמובן שהמקסימום במערך שגודלו 1 (מקרה הבסיס) הוא אותו איבר יחידי במערך.

```
int   max_array( int arr [] , int first_index , int n)        /* n > 0 */
{
  int tmp ;

  if(n == 1)
       return  arr [first_index] ;
  tmp=max_array(arr, first_index +1 ,n-1) ;     /* the recursive call */
  if(arr[first_index] > tmp )
       return  arr [first_index] ;
  return tmp ;
}
```

נראה איך הפונקציה תמצא את המקסימום במערך הבא :



4. חיבור וכפל שלמים (שאלה ממבחן) :

- בפקולטה למדעי הטבע ניבנה מחשב חדש עבור הסטודנטים. המחשב החדש מטפל רק במספרים טבעיים. על כן, במקום הטיפוסים הרגילים בשפה הוגדר טיפוס חדש - posint:

```
typedef        unsigned int  posint ;
```

הערה: את משמעות ביטוי זה נלמד בהמשך הקורס, כאשר נדון הוראות typedef. לעת עתה, ניתן לחשוב שישנו טיפוס יחיד בשפה המדוברת ששמו posint והוא מייצג מספרים טבעיים.
מתכנני השפה התרשלו בעבודתם ובמחשב החדש חסרות הפעולות : + , - , * , / , % .
כמו כן לא קיימות הלולאות : while , do-while ו - for .

סטודנט בר תושייה הבחין בתכונות הבאות של החיבור :

x + 0 = x .
x + y = (x + 1) + (y - 1) ;
והציע לממש באופן רקורסיבי תוך שימוש בפונקציות הבסיסיות, ++ ו – , פונקציה plus המקבלת שני פרמטרים מטיפוס posint ומחזירה את סכומם .

א: עליכם לכתוב את פונקצית plus בהתאם למגבלות הנ"ל.

ב: עליכם לכתוב פונקציה mult המשתמשת באמצעי רעיון, אלגוריתם ומשתמשת בפונקצית plus .

שאלה 7 (15 נקודות)

נתונה התכנית הבאה הכתובה בשפת C:

```c
#include <stdio.h>
void func(int num,int max)
{
    int i;
    if (num==0) return;
    for (i=0; i<num; i++)
        printf("*");
    for (; i<2*max-num; i++)
        printf(" ");
    for (; i<2*max; i++)
        printf("*");
    printf("\n");
    func(num-1,max);
    if (num!=1)
        printf("\n");
    for (i=0; i<num; i++)
        printf("#");
    for (; i<2*max-num; i++)
        printf(" ");
    for (; i<2*max; i++)
        printf("#");
}
main()
{
```

1) סדרת פיבונצ'י :

$a_0 = 0 , a_1 = 1 .$

$a_n = a_{n-1} + a_{n-2} \quad , \quad n > 1 .$

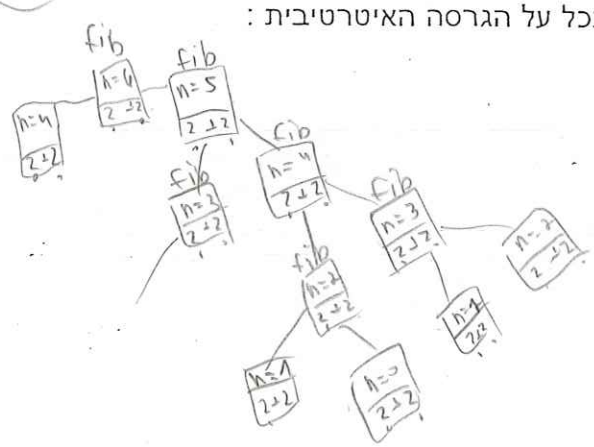
למשל הסדרה עד n=6 נראית כך :

0 , 1 , 1 , 2 , 3 , 5 , 8 . . . .

במקרה זה הסדרה מוגדרת באופן רקורסיבי ולכן נותר רק לכתוב את הקוד :

```
int  fib( int n)
{
    if(n <= 1)
     return n ;
    return fib(n-1) + fib(n-2) ;
}
```

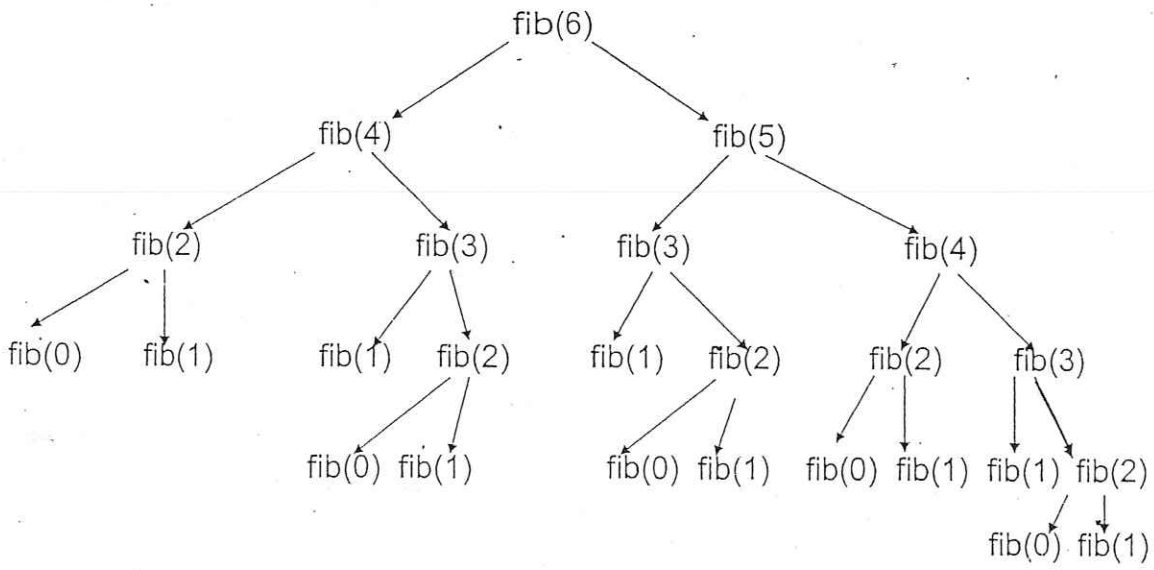
לשם השוואה נסתכל על הגרסה האיטרטיבית :




```
int fib_iterative(int n)
{
  int  x0 = 0 , x1 = 1 , tmp , i ;

   if(n<=1)
     return n ;
  for(i=2 ; i<=n ; ++i) {
     tmp = x0 + x1 ;
     x0 = x1 ;
     x1 = tmp ;
  }
 return  x1 ;
}
```

זכל על עץ הקריאות עבור fib(6) :



```c
/*****      chp1505.c    *****/

#include <stdio.h>
#include <string.h>


char *func (char *s)
{
    char t ;

    if (strlen(s) <= 1)
        return(s);

    t = *s ;

    strcpy (s, func(s+1));
    *(s+strlen(s)) = t ;

    return(s) ;
}


void main(void)
{
    printf ("... of ABC is %s", func("ABC"));
}
```

```c
/*****      chp1509.c       *****/

#include <stdio.h>
char *do_what (char *, int *);


char c ;

main()
{

    int i, array[25] ;
    char *ptr, c_array[]="AN EXAMPLE FOR RECURSION" ;
    for (i=0 ; i<25 ; ++i)
        array[i] = i ;

    ptr = do_what (c_array, array) ;
    printf("\nThe results are : %s and %d\n", ptr, c) ;
}


char *do_what (char *c_ptr, int *i_ptr)
{
    if (*c_ptr)
        *c_ptr = *(do_what(c_ptr+1, i_ptr+1)) ;
    else
        c = *c_ptr ;

    return(c_ptr - *i_ptr) ;
}
```

-1 [illegible]

-2 [illegible]

[illegible]

```
int plus( int x, int y)
{
    if( y==0)
        return x;
    return plus(y+x, --y )
}
```

[illegible]

```
int mult ( int x, int y)
{ if (y==0)
      return x;
  plus( x, x)
  return mult( x, --y);
```

```
if( y==0)
    return 0;
return plus(x, mult(x, --y));
```

חלק ד'

45 נקודות

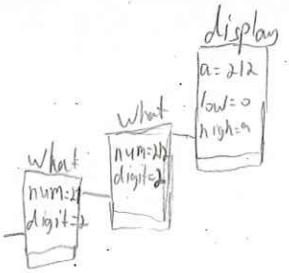
<u>ענה על 3 מתוך השאלות 5-8</u>

<u>שאלה 5</u>

```
int what (int num, int digit)
{
        if (num <10)
                if (num ==digit)
                        return 1;
                else
                        return 0;
        else
                return what(num /10,digit);
}

void display(int a[],int low,int high)
{
        if (low <=high )
        {
                if (what(a[low],a[low]%10)==1)
                        printf("%d  ",a[low]);
                display (a,low+1,high);
        }
        else
                printf("***");
}

void main()
{
        int a[]={212,123,2223,4,98,4554,546,22,19,5};
        display(a,0,9);
}
```



א. מה מבצעת הפונקציה what?

ב. מה פלט התוכנית, כתבי מעקב אחר ביצוע הפונקציה display.

ג. מה מבצעת התוכנית?

שאלה 6

נתונה התכנית הבאה הכתובה בשפת C:

```c
#include <stdio.h>

int func(int a,int b,int index)
{
   if (index <= 2) return 1;
   if (index == 3) return a+b;
   else return (func(b,a+b,index-1));
}

int main()
{
  int i;

  printf("\nThe numbers are: \n");
  for (i=1;i<=10;i++)
    printf("%d ",func(1,1,i));

  return 0;
}
```

func
a=1
b=2
index=4

func
a=1
b=1
index=5

The numbers are:
1
1
2
2

(8 נק') א. עבור הרצת func(1,1,5), תאר כמה פעמים נקראת הפונקציה עד שתחזיר את הערך.
רשום את ערכי הפרמטרים של ההרצות של func ואת הערך שהפונקציה func תחזיר לבסוף.

(7 נק') ב. מה תדפיס התכנית? 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

שאלה 8

א. מה הפלט? נמק!

ב. מה מבצעת התוכנית?

```
#define N 5
void fill(int arr1[], int arr2[], int mat[][N], int i, int j)
{
      if (i==0)
      {
            mat[i][j]=*arr1;
            mat[j][i]=*arr2;
      }
      else
            mat[i][j]=mat[i-1][j-1];
      if (j < N-1)
            fill(arr1+1,arr2+1,mat,i,j+1);
      else
            if(i<N-1)
                  fill(arr1,arr2,mat,i+1,1);
}
void main()
{     int mat[N][N];
      int arr1[N] = {1,2,3,4,5};
      int arr2[N] = {1,7,8,9,0};
      fill(arr1,arr2,mat,0,0);
      for(int i=0; i<N; i++)
      {
            for( int j=0; j<N; j++)
                  printf("%3d",mat[i][j]);
            printf("\n");
      }
}
```

fill i=0 j=0

fill i=0 j=1

fill i=0 j=2

i=0 j=3

i=0 j=4

i=1 j=1

i=1 j=2

i=1 j=2

i=1 j=3

i=1 j=4

אם

i=2 j=1

1	2	3	4	5
7	1	2	3	4
8	7	1	2	3
9	8	7	1	2
0	9	8	7	1

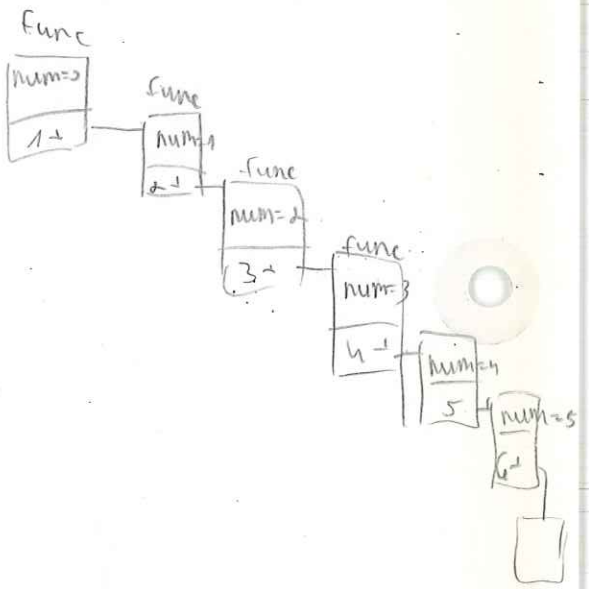
חלק ג' (45 נקודות)

ענה על שלוש מבין השאלות 5–8 (לכל שאלה – 15 נקודות).

שאלה 5

נתונה התכנית הבאה הכתובה בשפת C:

```
#include <stdio.h>
#define N 6
int func (int vec[],int num)
{
   int a;
   if (num>N-1)
      return 0;
   a=vec[num];
   if ( (num!=(N-1)))
      if ((vec[num]>vec[num+1]))
            num = N;
   return (a+func(vec,num+1));
}
int main()
{
   int vec[N];
   int res,i;
   for (i=0;i<N;i++)
      vec[i] = i+1;
   res = func(vec,0);
   printf("\n%d\n",res);
   for (i=3;i<N;i++)
      vec[i] = (3*i)%N;
   res = func(vec,0);
   printf("\n%d\n",res);
   return 0;
}
```



6

vec = 1, 2, 3, 4, 5, 6

21

vec = 1, 2, 3, 3, 0, 3

9

(5 נק') א. מה מבצעת הפונקציה func? סורקת את כל אברי המערך רק כשהם קטנים מאלה

(10 נק') ב. מה תדפיס התכנית?

כתוב במחברת את תכולת הווקטור vec לפני כל הרצת func (בכל פעם אחרי לולאת ה־for).