שאלה 1 – עצים בינאריים

נתונה ההגדרה הבאה:

```
typedef struct node{
      int val;
      struct node *left, *right;
}node;
```

כתוב פונקציה node *recreate_tree(int *in, int *post , int length) , אשר מקבלת כפרמטרים מצביע למערך של שלמים in שמכיל את ערכי סריקת עץ בינארי בשיטת inorder, מצביע למערך של שלמים post שמכיל את ערכי סריקת אותו העץ בשיטת postorder, ומספר האיברים שבעץ length. הפונקציה בונה מחדש את העץ הבינארי ומחזירה את שורשו.

תשובה:

```
node *recreate_tree(int *in, int *post, int length){
      node * root;
      int i;

      if (!length) return NULL;
      root = (node*) malloc (sizeof(node));
      root->val = post[length-1];
      for (i=0; in[i] != post[length-1]; i++);
      root->left = recreate_tree(in, post, i);
      root->right = recreate_tree(in+i+1, post+i, length-i-1);
      return root;
}
```

שאלה 2:

עיין בקטע הבא וסמן את כל התשובות הנכונות:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
void main()
{
        int *ptr, *arr[MAX];
        int i, j;
        for (i=MAX-1 ; i>=0; i--)
                if (arr[i] = (int *) malloc(i * sizeof(int)))
                        for (j=0; j<i; j++)
                                *(*(arr+i)+j) = j*i;
        ptr = *(arr+MAX-1);
        while (*ptr)
                printf ("%d ", *ptr--);
}
```

א. התכנית לא מדפיסה כלום.

ב. יש שגיאה בזמן ריצה (run time error).

ג. התכנית תדפיס: 9 18 27 36 45 54 63 72.

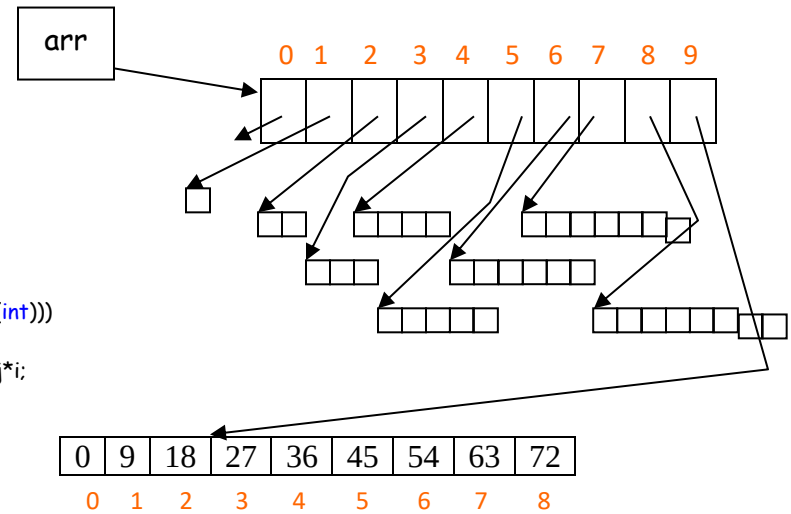
ד. התכנית תדפיס אינסוף אפסים.

ה. התכנית תדפיס 0.

ו. התכנית תדפיס ערכים לא ידועים.

ז. אף לא אחת מהתשובות לעיל.

תשובה:




```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
void main()
{
        int *ptr, *arr[MAX];
        int i, j;
        for (i=MAX-1 ; i>=0; i--)
                if (arr[i] = (int *) malloc(i * sizeof(int)))
                        for (j=0; j<i; j++)
                                *(*(arr+i)+j) = j*i;
        ptr = *(arr+MAX-1);
        while (*ptr)
                printf ("%d ", *ptr--);

}
```

א. התכנית לא מדפיסה כלום. – הסיבה היא ש- *ptr משתערך לאפס (ראה תרשים) וגוף לולאת ה- while לא מבוצע אף לא פעם אחת.

(II) איך ניתן לשנות את השורה: ptr = *(arr+MAX-1); ע"מ שתשובה ג' תיהיה נכונה?

<u>פתרון</u>: צריך לשנות ל- ptr = *(arr+MAX-1)+8;

שאלה 3 (15 נקודות)

נתונה ההגדרה הבאה:

```
typedef struct Node {
    int numOfNbrs;
    struct Node* neighbors[1000];
    int color;
    int tmp;   // שדה עזר
} Node;
```

נתון מבנה נתונים הבנוי מאוסף של צמתים מטיפוס ***Node***, כך שכל צומת מכיל מערך של מצביעים לשכנים שלו.

שכנות במבנה נתונים זה מוגדרת כדו-כיוונית. כלומר: אם X הוא שכן של Y, אזי Y שכן של X גם כן.

השדה ***numOfNbrs*** מייצג את מספר השכנים שיש לצומת. ניתן להניח שכל השכנים נמצאים בתחילתו של המערך ***neighbors*** וכן שלא יהיו יותר מ1000 שכנים לצומת.

כל צומת מכיל גם שדה ***color*** המייצג את הצבע של הצומת.

בנוסף לשדות לעיל" בכל צומת קיים שדה עזר ***tmp*** שמאותחל ל-0 ניתן לעשות בו שימוש כרצונכם.

צביעה חוקית של מבנה הנתונים הנ"ל היא קביעת כל ערכי הצבעים כך שכל צומת צבוע בצבע שונה משכניו.

השלימו את הפונקציה ***int legalColoring(Node* n)*** אשר מקבלת צומת מסוים ***n*** ומחזירה 1 אם מבנה הנתונים צבוע באופן חוקי ו-0 אחרת.

מותר להשתמש בשדה ***temp*** כרצונכם.

השלימו את הקטעים החסרים (המסומנים ב- **?? N ??**) בקוד הבא.

```
int legalColoring(Node* n){
    int i;

    if ( ?? 1 ?? )
          ?? 2 ?? ;
    ?? 3 ?? ;
    for (i=0; ?? 4 ?? ; i++)
          if (?? 5 ??  || ?? 6 ??)
               ?? 7 ?? ;
    return ?? 8 ?? ;
}
```

פתרון:

```
int legalColoring(Node* n){
      int i;

      if ( n->tmp )
            return 1 ;

      n->tmp = 1 ;

      for (i=0; i<n->numOfNbrs ; i++)
            if (n->color == n->neighbors[i]->color ||
                        !legalColoring(n-> neighbors[i]))
                  return 0 ;
      return 1;

}
```