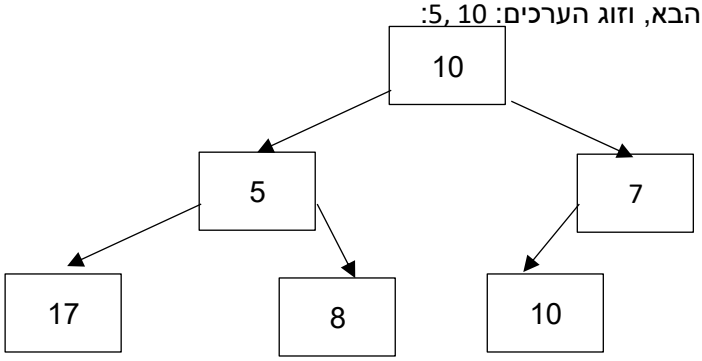
ונבנה עץ בינארי (שאינו בהכרח עץ חיפוש).

<u>סעיף א'</u> (45 נ')
כתבו פונ' אשר מקבלת מצביע לשורש העץ, וכן שני ערכים שלמים: min, max.
על הפונ' להחזיר מצביע לתת-העץ הגדול ביותר (כלומר זה שכולל את הכמות הגדולה ביותר של צמתים) שכל הערכים שבו גדולים או שווים מ: min, וקטנים או שווים מ: max.

<u>הערות:</u>
א. אם לא נמצא תת-עץ כנדרש יש להחזיר את הערך NULL.
ב. ראוי שהפונ' שתכתבו תקבל פרמטרים בדיוק כפי שמתואר בשאלה. הפונ' רשאית לזמן כל פונ' שהיא, ולהעביר לה פרמטרים כפי שיקול דעתכם.
ג. הקפידו על היעילות.

<u>לדוגמה</u>: עבור העץ הבא, וזוג הערכים: 5, 10:



יוחזר מצביע לצומת של 7.
לו הערך בצומת של 17 (העלה השמאלי) היה 9 (ולא 17) אזי היה מוחזר מצביע לצומת של 5.
לו *בעץ המצויר*, בצומת של 7 היה מצוי הערך 11 אזי היה מוחזר מצביע לצומת של 8 או לזה של 10.

```
const struct Node * count_nodes_between(const struct Node * root,
                                        int min,
                                        int max,
                                        int & max_count) {
    int max_left = 0;
    int max_right = 0;
    const struct Node * left = NULL;
    const struct Node * right = NULL;

    //Leaf
    if(root->_left == NULL && root->_right == NULL) {
        if(root->_data >= min && root->_data <= max) {
            max_count++;
            return root;
        } else {
            return NULL;
        }
    }
```

```
    if(root->_left != NULL)
        left = count_nodes_between(root->_left, min, max,
max_left);
    if(root->_right != NULL)
        right = count_nodes_between(root->_right, min, max,
max_right);

    if(max_left == 0 && max_right == 0) {
        return NULL;
    }

    if(root->_data >= min && root->_data <= max &&
       left == root->_left && right == root->_right) {
        max_count = 1 + max_left + max_right;
        return root;
    }

    if(max_left > max_right) {
        max_count = max_left;
        return left;
    } else {
        max_count = max_right;
        return right;
    }
}
```

אופציה ב

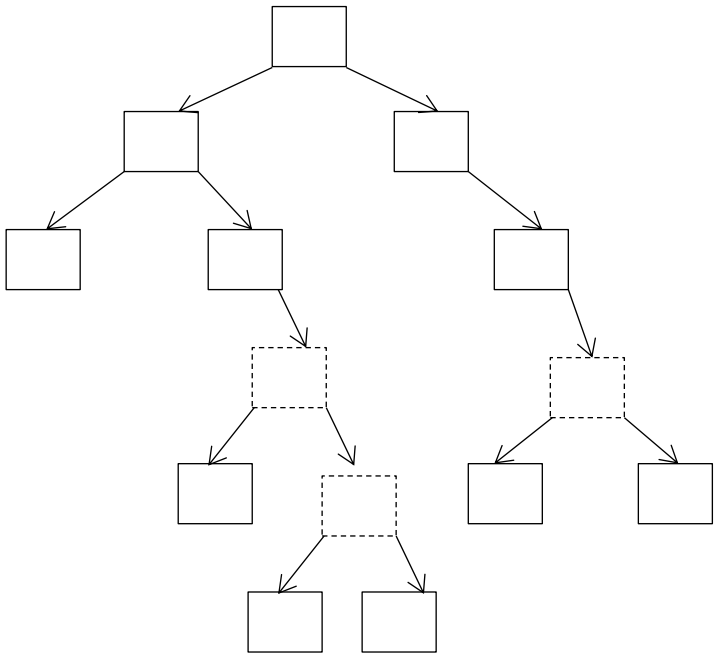
```
------------------------------ 1 ------------------------------
struct Node *min_to_max_node(const struct Node *root,
                                                    int min,
                                                    int max)
{
       int max_nodes_counter = 0;
       struct Node *max_sub_tree;

       check_sub_tree(root, min, max, max_sub_tree,
max_nodes_counter);

       if(max_nodes_counter == 0)
              return NULL;

       return max_sub_tree;
}
//----------------------------------------
void check_sub_tree(const struct Node *root,
                                     int min,
                                     int max,
                                     struct Node *&max_sub_tree,
```

```
                                        int &max_nodes_counter)
{
        if(root == NULL)
                return;
        int nodes_counter = 0;

        if(check_min_max(root, min, max, nodes_counter)
                if(max_nodes_counter < nodes_counter)
                {
                        max_nodes_counter = nodes_counter;
                        max_sub_tree = root;
                }
        check_sub_tree(root->_left, min, max, max_sub_tree,
max_nodes_counter);
        check_sub_tree(root->_right, min, max, max_sub_tree,
max_nodes_counter);
}
//----------------------------------------
bool check_min_max(const struct Node *root,
                                        int min,
                                        int max,
                                        int &nodes_counter)
{
        if(root == NULL)
                return true;

        if(root->_data >= min && root->_data <= max)
        {
                nodes_counter++;
                if(check_min_max(root->_left, min, max,
nodes_counter)
                        if(check_min_max(root->_right, min, max,
nodes_counter)
                                return true;
        }
        nodes_counter = 0;
        return false;
}
------------------------------ 2 ------------------------------
```

סעיף א' (45 נ')
כתבו פונ' המקבל מצביע לשורש העץ, ומחזירה כמה צמתים בעץ מקיימים ש:

א. לצומת יש שני ילדים (לא ריקים).
ב. הצומת מצוי בתוך תת-עץ כך שלצומת בשורש תת-העץ יש רק ילד אחד (לא ריק), במלים אחרות: לצומת עם שני הילדים יש אבא, סבא, סבא רבה,... לפחות אחד שיש לו רק ילד יחיד (אין צורך שלכל אבותיו יהיה ילד יחיד, רק לאחד מאבותיו יהיה ילד יחיד).

לדוגמה: עבור העץ הבא



יוחזר הערך שלוש: שכן שלושת הצמתים שמצוירים בקו מקווקו מקיימים את התנאי: יש להם שני ילדים, והם מצויים בתוך תת-עץ שלשורשו יש ילד יחיד

סעיף ב' (5 נ')
הסבירו מהו זמן הריצה של הפונ' שכתבתם

```
unsigned get_2child_from_1child(const struct Node *root)
{
	unsigned has_2child ;
	return get_2child_from_1child(root, has_2child) ;
}
//*************************************************************************


unsigned get_2child_from_1child(const struct Node *root,
						unsigned &has_2child)
{
	if (root == NULL)
		return NULL ;

	unsigned has_2child_left = 0, has_2child_right = 0 ;

	unsigned has_2child_from_1child_left =
		get_2child_from_1child(root->_left, has_2child_left) ;

	unsigned has_2child_from_1child_right =
		get_2child_from_1child(root->_right, has_2child_right) ;

	has_2child = has_2child_left + has_2child_right ;
	unsigned has_2child_from_1child =
		has_2child_from_1child_left + has_2child_from_1child_right ;

	if (root->_left != NULL && root->_right != NULL)
		has_2child ++ ;
	else if (root->_left != NULL || root->_right != NULL)
	{
		has_2child_from_1child += has_2child ;
		has_2child = 0 ;
	}
	return has_2child_from_1child ;
}
```

ונבנה עץ בינארי של נקודות. לשם הפשטות נניח שהנקודות בעץ אינן מצויות על הצירים.

כתבו פונ' המקבלת מצביע לשורשו של העץ, ומחזירה מצביע לתת-העץ הגדול ביותר המקיים שכל הנקודות בתת-העץ מצויות באותו רביע. באמצעות שני פרמטרי הפניה יוחזרו:

א. מספרו של הרביע.
ב. מספר הצמתים בתת-העץ.

<u>הערות</u>:
אתם רשאים להניח שבתכנית שלכם מוגדרת הפונ':

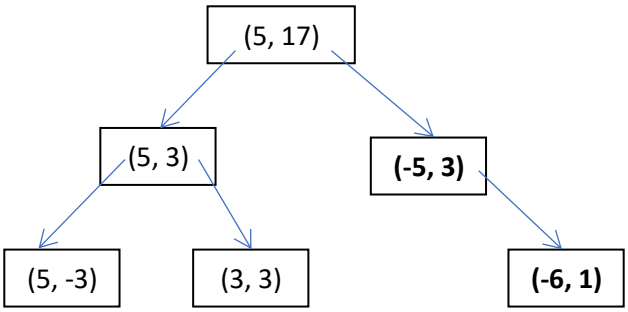
```
int quarter(const struct Node *root) ;
```

הפונ' מקבלת מצביע לצומת ומחזירה את מספרו של הרביע בו מצויה הנקודה root->_data.
אם root==NULL אזי מוחזר הערך אפס.

הפונ' שתכתבו רשאית להחזיר את המצביע לשורשו של תת-העץ המבוקש לא כערך החזרה, אלא באמצעות פרמטר הפניה.

הפונ' שתכתבו יכולה לקבל פרמטרים כפי שיקול דעתכם.

<u>לדוגמה</u>: אם הפונ' תקרא עם העץ:



אזי יוחזר מצביע לצומת של (-5, 3) והמידע כי בתת-עץ זה יש שני צמתים, ושניהם מצויים ברביע השני.
לו בעלה המכיל את: (5, -3) הייתה מצויה במקום נקודה זאת, הנקודה: (5, 2) אזי היה מוחזר מצביע לצומת של (5, 3), והמידע כי בתת-העץ יש שלושה צמתים, המצויים ברביע הראשון.

```
int sub_same_quarter(const struct Node *root,
                                      int &max_counter,
                                      int &max_quarter,
                                      const struct Node
*&max_node)
{
    int left, right;

    if(root == NULL)
        return 0;

    if(root->_left == NULL && root->_right == NULL)
       {
        if(max_counter < 1)
               {
            max_counter = 1;
            max_node = root;
            max_quarter = quarter(root);
        }
        return 1;
    }

    left = sub_same_quarter(root->_left, max_counter,
max_quarter, max_node);
    right = sub_same_quarter(root->_right, max_counter,
max_quarter, max_node);

    if((left != 0 || right != 0)&&
        same_q(root, root->_left)&&
        same_q(root, root->_right))
       {
        if(right + left + 1 > max_counter)
               {
            max_counter = right + left + 1;
            max_node = root;
            max_quarter = quarter(root);
            return right + left + 1;
        }
    }
    return 0;
}
//----------------------------------------
bool same_q(const struct Node * root,
                      const struct Node * child)
{
    if(child == NULL)
        return true;

    if(quarter(root) == quarter(child))
        return true;

    return false;
}
```

```
//----------------------------------------
int quarter(const struct Node * root)
{
    if(root == NULL)
        return 0;
    if(root->_data._x > 0 && root->_data._y > 0)
        return 1;
    if(root->_data._x < 0 && root->_data._y > 0)
        return 2;
    if(root->_data._x < 0 && root->_data._y < 0)
        return 3;
    return 4;
}
```

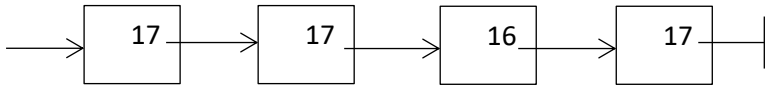
בתכנית הוגדר:

```
struct Node {
  int _data ;
   struct Node *_next ;
}
```

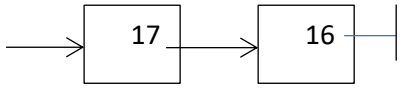
ונבנתה רשימה מקושרת.

<u>סעיף א':</u> (43 נ')
כתבו פונ' המקבלת מצביע לראשה של הרשימה המקושרת, ומסירה מהרשימה איברים כפולים,
כלומר בתום ביצוע הפונ' כל ערך יופיע ברשימה פעם אחת בלבד.

לדוגמה: אם הפונ' תקבל את הרשימה הבאה



אזי בתום ביצוע הרשימה תראה:



<u>הערה\רמז:</u> כתבו פונ' קצרה ככל שניתן מבחינת אורך הקוד שלכם

<u>סעיף ב':</u> (7 נ')
הסבירו מה יהיה זמן הריצה של הפונ' שכתבם בסעיף א'

```
struct Node ** delete_duplicad(struct Node **p2head)
{

	struct Node **new_head = p2head;

	while( (*p2head != NULL)  && (*p2head)->_next != NULL)
	{
		if( (*p2head)-> data !=  (*p2head) ->_next->data )
			p2head = &((*p2head)->_next);
		else
		{
			struct Node *temp =  (*p2head)->_next;
			(*p2head)->_next= (*p2head)->_next ->_next);

			delete  temp;
		}

	}

	return new_head;
}
```