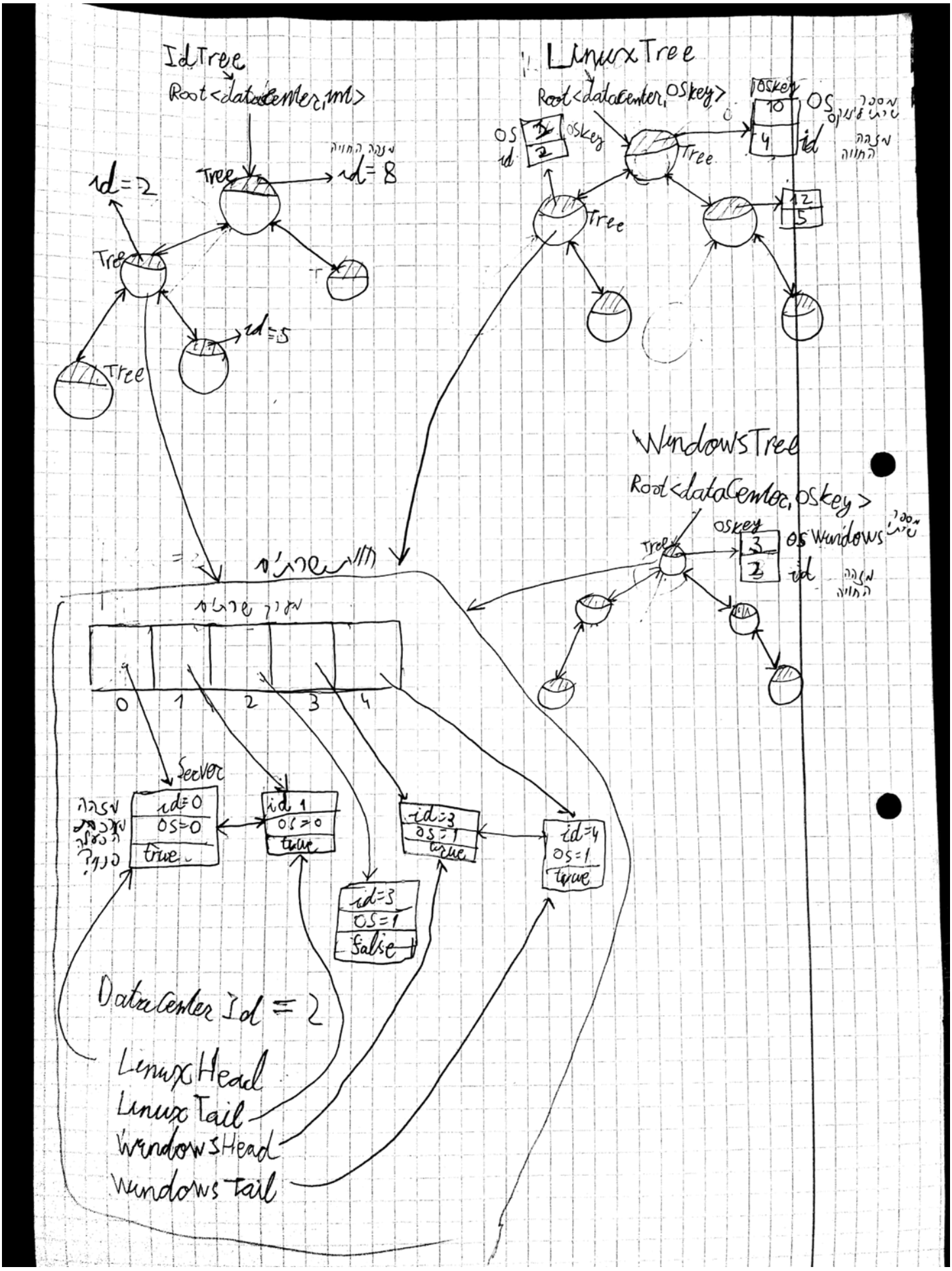
IdTree
Root<dataCenter, int>
Tree
id=2
id=8
id=5
LinuxTree
Root<dataCenter, OSkey>
OSkey
OS
id
WindowsTree
Root<dataCenter, OSkey>
OS Windows
מערך שרתים
0
1
2
3
4
Server
id=0
OS=0
true
id=1
OS=0
true
id=3
OS=1
false
id=3
OS=1
true
id=4
OS=1
true
DataCenter Id = 2
LinuxHead
LinuxTail
WindowsHead
WindowsTail

תיאור מבנה הנתונים

שרת - Server

מייצג שרת. מכיל את ה ID של השרת שהוא מייצג, סוג מערכת ההפעלה שמותקנת עליו וגם שומר משתנה בוליאני שאומר אם השרת פנוי או לא.

חוליה - Node

מכיל מצביע לserver שהוא מייצג, ומכיל מצביעים לאיברים שלפניו ואחריו ברשימה.

חוות שרתים - Data center

מייצג חוות שרתים. מכיל שתי רשימות מקושרות:

רשימה אחת מכילה nodeים של שרתים חופשיים של לינוקס – כל node ברשימה

מייצג שרת חופשי מסוג לינוקס.

רשימה אחרת מכילה nodeים של שרתים חופשיים של windows – כל node ברשימה

מייצג שרת חופשי מסוג windows.

נחזיק במבנה מצביעים לראש וזנב של שתי הרשימות האלו(כלומר האיבר הראשון, והאיבר האחרון).

בנוסף נחזיק מערך שיכיל nodeים של כל השרתים שקיימים בחוות השרתים הנוכחית כדי שנוכל לגשת לכל node בזמן O(1).

בנוסף מבנה זה מכיל את ה ID של חוות השרתים הנוכחית, מספר השרתים מסוג לינוקס,

מספר השרתים מסוג ווינדוס וסה"כ מספר השרתים בחווה.

מערכת חוות שרתים - DataCenterManager

המבנה הראשי. מבנה זה יכיל 3 עצי חיפוש AVL. כל עץ יכיל מצביע למפתח ומצביע ל-Data. בנוסף למצביעים לבנים, כל צומת יצביע לאב שלו, כדי להקל על הניווט בעץ. בנוסף נחזיק שדה height שיסמן את גובה העץ.

<u>עץ ראשון</u> – כל צומת יכיל כמפתח את ה ID של חוות השרתים שאותה הוא מייצג,

וכ- data יכיל מצביע לחוות השרתים שאותה הוא מייצג. נקרא לו IDTree.

<u>עץ שני</u> – כל צומת יכיל כמפתח את מספר שרתי linux וגם ID של חוות השרתים שאותה הוא מייצג, וכ- data יכיל מצביע לחוות השרתים שאותה הוא מייצג. נקרא לו linuxTree.

<u>עץ שלישי</u> – כל צומת יכיל כמפתח את מספר שרתי windows וגם ID של חוות השרתים שאותה הוא מייצג, וכ- data יכיל מצביע לחוות השרתים שאותה הוא מייצג. נקרא לו windowsTree.

העצים linuxTree ו windowsTree יהיו ממוינים בסדר יורד על פי מספר השרתים מסוג שלהם(windowsTree לפי windows ובאופן דומה לגבי לינוקס) שיש בחווה. במקרה של שוויון נמיין אותם על פי המזהים בסדר עולה.

המפתחות מכילים את מספר שרתי מערכת ההפעלה מהסוג הנדרש וגם ID ולכן יש את כל המידע הנחוץ לביצוע המיון הנדרש בפונקציה GetDataCentersByOS.

מימוש הפעולות

דרוש מבנה נתונים למימוש הפעולות הבאות:

void * Init()

מאתחל מבנה נתונים ריק.

פרמטרים: אין.

ערך החזרה: מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון.

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

הפעולה מאתחלת את מבנה הנתונים. הפעולה בונה את מערכת חוות השרתים ומאתחלת שלושה עצים מתאימים ריקים. גודל המבנה קבוע ולכן הבנייה אורכת O(1) זמן ו- O(1) מקום.

StatusType AddDataCenter(void *DS, int dataCenterID, int numOfServers)

הוספת חוות שרתים חדשה עם מזהה *dataCenterID* שיש בה *numOfServers* שרתים פנויים וממוספרים במספרים עוקבים החל מ-0 (0,1,2,...) וכן הלאה). מערכת ההפעלה של כל השרתים החדשים היא Linux כערך ברירת מחדל.

פרמטרים:

DS	מצביע למבנה הנתונים.
dataCenterID	מזהה חוות השרתים.
numOfServers	מספר השרתים שיש בחווה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL, numOfServers<=0 או dataCenterID<=0.
FAILURE	אם כבר קיימת חווה עם המספר המזהה הנתון.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(log(n)+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים במערכת ו-m הוא numOfServers.

1. נערוך חיפוש בעץ IDTree לID שקיבלנו. חיפוש זה עורך O(logn) זמן. אם קיימת חוות שרתים עם אותו מזהה אז לא נוסיף אותה לעץ אלה נחזיר FAILURE.
2. ניצור חוות שרתים חדשה – סיבוכיות זמן O(m), כי נכריז על מערך באורך m ונעבור איבר איבר ונאתחל אותו.
3. נאתחל את הרשימות ואת המשתנים. באתחול כל השרתים מסוג לינוקס וכולם פנויים ולכן הרשימה שמייצגת את שרתי הלינוקס הפנויים תכיל את כל השרתים שיש בחווה, והרשימה של שרתי ווינדוס הפנויים תהיה ריקה.
4. נוסיף את חוות השרתים לשלושת העצים. הוספת החווה לכל עץ לוקחת O(log n) זמן כפי שלמדנו בהרצאות.

- סה"כ סיבוכיות זמן O(log n + m) וסיבוכיות מקום O(m).

```
StatusType RemoveDataCenter(void *DS, int dataCenterID)
```

מחיקת חוות השרתים עם המזהה *dataCenterID* וכל השרתים שנמצאים בה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL או dataCenterID<=0.
	FAILURE	אם לא קיימת חווה עם המספר המזהה הנתון.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log(n)+m)$	במקרה הגרוע, כאשר n הוא מספר חוות השרתים במערכת ו-m הוא מספר השרתים בחווה (m הוא 0 אם אין חווה עם המזהה הנתון).

נבצע מחיקה משלושת העצים תוך שימור תכונות העץ. כפי שלמדנו בהרצאות, האלגוריתם מתבצע בזמן O(log n) לכל עץ, כלומר סה"כ סיבוכיות הזמן: O(3log n) = O(log n)

```
StatusType RequestServer(void *DS, int dataCenterID, int serverID, int os, int *assignedID)
```

הקצאת השרת *serverID* שבחווה *dataCenterID* והתקנת מערכת ההפעלה *os* עליו.

אם השרת *serverID* תפוס אז המערכת תקצה שרת פנוי אחר שכבר מותקנת עליו מערכת ההפעלה המבוקשת.

אם לא קיים כזה אז המערכת תקצה שרת פנוי עם מערכת הפעלה שונה ותתקין עליו את מערכת ההפעלה המבוקשת.

השרת שהוקצה ייחשב כתפוס ולא יהיה ניתן להקצות אותו שוב עד שישוחרר (באמצעות FreeServer). בנוסף, יש להחזיר את מזהה השרת שהוקצה באמצעות הפוינטר *assignedID*.

יש חשיבות לאופן בחירת השרת שמחזירים במידה והשרת המבוקש תפוס. בהמשך התרגיל יש הסבר לסדר הנדרש ודוגמה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
	serverID	מזהה השרת הנדרש.
	os	סוג מערכת ההפעלה (Linux=0 ו-Windows=1).
	assignedID	מצביע למשתנה אשר מזהה השרת שהוקצה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, serverID<0, serverID>=numOfServers, dataCenterID<=0, os<0 או os>1 או assignedID==NULL. (כאשר numOfServers הוא מספר השרתים בחווה המבוקשת)
	FAILURE	אם לא קיימת חווה עם המזהה המבוקש או שלא נמצא בה אף שרת פנוי.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log(n))$	במקרה הגרוע, כאשר n הוא מספר חוות השרתים.

1. נבצע בדיקה של תקינות הארגומנטים שיתקבלו – אם לפחות אחד מהם לא תקין נחזיר INVALID INPUT.

בדיקה זו תתבצע בסיבוכיות זמן של O(1).

2. נחפש את חוות השרתים עם מזהה dataCenterID על ידי הפעלת אלגוריתם חיפוש בעץ – כפי שלמדנו בהרצאה זמן ריצה שלו O(log n)
3. אם לא קיימת חווה עם המזהה dataCenterID או אם שדות ה linuxHead ו windowsHead שווים ל nullptr זה אומר שאין שרתים פנויים בחווה ולכן נחזיר FAILURE.
אחרת, קיימת חווה עם המזהה הרצוי וגם יש בה לפחות שרת פנוי אחד. נשמור על מספר השרתים מסוג linux ומספר השרתים
מסוג windows בשתי משתנים.
נלך לשרת בעל מזהה serverID ונבדוק אם הוא חופשי. אם כן – נוציא אותו מהרשימה של השרתים החופשיים שבא הוא נמצא, נעדכן את שדה ה isFree ל FALSE ונעדכן את מערכת ההפעלה של השרת לזו שקיבלנו כארגומנט(os), נכניס את הערך של מזהה השרת במשתנה
assignedID* ונוציא את הצמתים הרלוונטים מעץ windowsTree ו linuxTree ונחזיר אותם בחזרה לאחר עדכון שדות(בכך ששמרנו את מספר שרתי לינוקס ווינדוס הישנים יכולנו למצוא את הצמתים הרלוונטים להסרה ולאחר עדכון הכנסנו בחזרה ובכך שמרנו על המבנה) ונחזיר SUCCESS.זמן ריצה של פעולות אלה הוא O(log n)

4. אחרת השרת לא חופשי, ולכן נבצע את הפעולות הבאות:

אם מערכת ההפעלה המבוקשת היא linux – ניגש לשדה linuxHead ואם הוא לא nullptr אז קיים לפחות סרבר חופשי אחד מסוג זה – נוציא את linuxHead מהרשימה, נעדכן את שדה ה isFree שלו
להיות FALSE, ונעדכן את assignedID* לקבל את ערך מזהה השרת שnode שהוצאנו מייצג ונחזיר SUCCESS.
אחרת, linuxHead הוא nullptr כלומר לא קיים שרת מסוג לינוקס שהוא חופשי בחוות שרתים הזו.
כי בהכרח קיים לפחות שרת אחד חופשי(כי אחרת האלגוריתם היה עוצר בשלב 3) – לכן בהכרח
windowsHead לא שווה ל nullptr כלומר יש לפחות שרת פנוי אחד מסוג windows.
נוציא מרשימה זו את ראש הרשימה – נעדכן את שדה isFree שלו להיות FALSE ונעדכן את
assignedID* להיות שווה לערך מזהה השרת שה node שהוצאנו מייצג ונוציא את הצמתים הרלוונטים משלושת העצים ונחזיר אותם בחזרה לאחר עידכון השדות ונחזיר SUCCESS.

אם מערכת ההפעלה המבוקשת היא windows אז נבצע את אלגוריתם 4 רק שבמקום linux
יהיה windows ובמקום linuxHead יהיה windowsHead וכך הלאה.
זמן ריצה של אלגוריתם זה הוא O(log n).

- סה"כ זמן ריצה של כל פעולות אלו הוא O(log n)

```
StatusType FreeServer(void *DS, int dataCenterID, int serverID)
```

שחרור השרת *serverID* שבחווה *dataCenterID*. לאחר הפעולה, השרת יהיה פנוי ותישאר עליו מערכת ההפעלה האחרונה שהותקנה עליו.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
	serverID	מזהה השרת הנדרש.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, serverID<0, serverID>=numOfServers או dataCenterID<=0. (כאשר numOfServers הוא מספר השרתים בחווה המבוקשת)
	FAILURE	אם לא קיימת חווה עם המזהה הנתון או שהשרת עם המזהה הנתון כבר פנוי.
	SUCCESS	במקרה של הצלחה.
<u>סיבוכיות זמן</u>:	$O(log(n))$	במקרה הגרוע, כאשר n הוא מספר חוות השרתים.

1. נבצע בדיקה של תקינות הארגומנטים חוץ מהתנאי (serverID >= numOfServers) – אם לפחות אחד מהם לא תקין נחזיר INVALID INPUT. בדיקה זו תתבצע ב(O(1 .
2. נבצע חיפוש של חוות השרתים בעלת מזהה dataCenterID. אלגוריתם החיפוש הוא כמו שלמדנו בהרצאה מתבצע ב (O(log n. אם היא לא קיימת – נחזיר FAILURE.
3. נבדוק שמתקיים serverID >= numOfServers. אם לא מתקיים נחזיר INVALID INPUT.
4. אחרת, חוות השרתים קיימת ו-serverID תקין. נשמור את מספר שרתי לינוקס ווינדוס וניגש לשרת בעל מזהה serverID שבחווה זו, אם הוא פנוי נחזיר FAILURE.
אחרת הוא לא פנוי ולכן נבדוק את סוג מערכת ההפעלה שמותקנת עליו, נשנה את שדה ה isFree שלו ל TRUE ונכניס את ה node לסוף הרשימה שמתאימה למערכת ההפעלה שמותקנת עליו.(כלומר אם מסוג linux ה node יהפוך ל linuxTail החדש וכנ"ל לגבי windows).
נוציא את הצמתים מ linuxTree ו windowsTree – בזכות זה ששמרנו את מספר שרתי ווינדוס ולינוקס לפני השינוי זה מתאפשר כי הם ערכי המפתחות בשני העצים.
לאחר העדכון נחזיר
את הצמתים בחזרה עם הערכים המעודכנים.
נחזיר SUCCESS.
רצף פעולות זה יתבצע ב (O(log n.

- סה"כ זמן ריצה הוא (O(log n.

```
StatusType GetDataCentersByOS(void *DS, int os, int **dataCenters, int* numOfDataCenters)
```

החזרת כל המזהים של חוות השרתים ממוינים בסדר יורד על פי מספר השרתים מסוג *os* (לא משנה אם השרת תפוס או פנוי) שיש בחווה. במקרה של שוויון יש למיין מיון משני על פי המזהים בסדר עולה.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	os	סוג מערכת ההפעלה (Linux=0 ו-Windows=1).
	dataCenters	מצביע למערך אשר יכיל את מזהי חוות השרתים.
	numOfDataCenters	מצביע למשתנה אשר יכיל את כמות המזהים המוחזרים.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, dataCenters==NULL, numOfDataCenters==NULL, os<0 או os>1.
	FAILURE	אם אין במבנה חוות שרתים בכלל.
	SUCCESS	במקרה של הצלחה.
<u>סיבוכיות זמן</u>:	$O(n)$	במקרה הגרוע, כאשר n הוא מספר חוות השרתים.

שימו לב שאתם מקצים את המערך בגודל המתאים, כמו כן אתם צריכים להקצות את המערך בעצמכם באמצעות malloc (כי הוא ישוחרר בקוד שניתן לכם באמצעות free).

1. נבצע בדיקה של תקינות הארגומנטים של הפונקציה – אם לפחות אחד מהם לא תקין נחזיר
INVALID INPUT. בדיקה זו תתבצע ב(O(1 .
2. נפעיל את אלגוריתם getLength של העץ ונספור את מספר הצמתים שיש בשורש.
נשים את התוצאה במשתנה numOfDataCenters*.
אם האורך 0 אז העץ ריק שזה אומר שאין במבנה חוות שרתים ולכן נחזיר FAILURE.
סיבוכיות זמן: O(n).
3. נאתחל את המערך שקיבלנו כארגומנט להיות מאורך מספר חוות השרתים שיש במבנה – כאשר המערך יכיל את מזהי חוות השרתים.
סיבוכיות זמן: O(1).
4. נפנה לעץ הרלוונטי כתלות ב os שקיבלנו כארגומנט(אם לינוקס אז לפי עץ לינוקס וכו).
אתחל מערך עזר שיכיל חוות שרתים באורך גודל העץ(לכל העצים אותו האורך). נעבור על העץ ונכניס איברים למערך לפי סדר מפתחות יורד. כזכור, המפתחות בעץ ממויינים בצורה שמתאימה לדרישות הפעולה.
סיבוכיות זמן : O(n).
5. נעתיק את חוות השרתים ממערך זה למערך dataCenters*.
זה יקח O(n) זמן.

- סה"כ
- סיבוכיות זמן : O(n)
- סיבוכיות מקום: O(n)

void Quit(void **DS)

הפעולה משחררת את המבנה. בסוף השחרור יש להציב ערך NULL ב-DS, אף פעולה לא תקרא לאחר מכן

פרמטרים: DS מצביע למבנה הנתונים.

ערך החזרה: אין.

סיבוכיות זמן: $O(n+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים ו-m הוא מספר השרתים הכולל בכל החוות.

סיבוכיות מקום (עבור המבנה וכל הפעולות) - $O(n+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים ו-m הוא מספר השרתים הכולל בכל החוות.

1. עוברים כל חוות השרתים דרך העץ של IDTree ומשחררים את השרתים שם, לאחר מכן משחררים את IDTree.
   סיבוכיות זמן: O(m)
2. אז עוברים דרך שתי העצים הנותרים ומשנים את איבר ה data ל nullptr(כי שיחררנו את הdata בעץ הראשון).
   סיבוכיות זמן: O(n)
3. לאחר מכן משחררים את שני העצים, משחררים את המבנה ואז נשים NULL ב DS*.
   סיבוכיות זמן: O(n)

- סה"כ סיבוכיות זמן: O(n + m)