1. מבני הנתונים הבסיסים :

:Tree עץ AVL גנרי התומך בפעולות – חיפוש, הכנסה, הוצאה וסיור, בנוסף נחזיק מונה שישמור את מספר האיברים בעץ.

מבני הנתונים בו השתמשנו:

- Companies: עץ AVL המכיל אובייקטים מטיפוס Company. יחזיק בתוכו את כל החברות במערכת בסדר ממוין לפי ה-CompanyID.
- NonEmptyCompanies: עץ AVL המכיל אובייקטים מטיפוס Company. יחזיק בתוכו רק חברות בעלות עובד אחד לפחות. ממוין לפי ה- CompanyID. נשים לב שבעץ זה יש לכל היותר כמות צמתים ככמות העובדים בכל המערכת. לכן פעולות עליו יהיו לכל היותר (או עבור סיורים)
- EmployeesByID: עץ AVL המכיל אובייקטים מטיפוס Employee. יחזיק בתוכו את כל העובדים במערכת בסדר ממוין לפי ה-. EmployeeID
- EmployeesBySalary: עץ AVL המכיל אובייקטים מטיפוס Employee. יחזיק בתוכו את כל העובדים במערכת בסדר ממוין לפי ה- Salary ואם יש 2 עובדים עם אותה משכורת, נבדיל לפי ה.ID
- בטיפוס של Company יש שדה employeesBySalary המייצג עץ AVL המכיל את כל עובדיי החברה ממויינים לפי Salary של כל עובד .
- בנוסף, בטיפוס של Company יש שדה employeesByID המייצג עץ AVL המכיל את כל עובדיי החברה ממוניים לפי ה IDשל כל עובד.
- recordEarner: מצביע לעובד המרוויח ביותר בכל המערכת.

מבנה הנתונים שלנו יכיל את הטיפוסים הבאים:

Company	Employee
int id	int id
int value	shared_ptr<Company> employer
AVLTree<shared_ptr<Employee>> employeesByID	
AVLTree<shared_ptr<Employee>> employeesBySalary	int salary
shared_ptr<Employee> highestEarner	int grade

AddCompany - בעת הוספת חברה חדשה נבצע את השלבים הבאים:

1. נחפש את המפתח companyID בעץ ,Companies אם קיים נזרוק חריגה $O(\log k)$
2. ניצור טיפוס חדש מסוג Company עם ה ID-וה value-הנתונים ועצי עובדים לפי מזהה ולפי משכורת ריקים $O(1)$
3. נכניס את הטיפוס החדש במקום המתאים בעץ $O(\log k)$ Companies
4. ניצור עצים ריקים עבור EmployeesBySalary ו-$O(1)$ EmployeesByID

סה"כ – סיבוכיות זמן : $O(\log k)$

AddEmployee - בעת הוספת עובד חדש נבצע את השלבים הבאים:

1. נחפש את המפתח EmployeesID בעץ ,EmployeesByID אם קיים נזרוק חריגה $O(\log n)$
2. נחפש את המפתח CompanyID בעץ ,Companies אם לא קיים נזרוק חריגה $O(\log k)$
3. נקצה טיפוס חדש מסוג Employee עם ה ID-ה salary-וה grade-הנתונים $O(1)$
4. אם החברה שמצאנו בסעיף 2 לא מכילה עובדים, נוסיף אותה ל $O(\log n)$,NonEmptyCompanies
5. נעדכן את שדה ה employerלהצביע על החברה שמצאנו בסעיף 2. $O(1)$
6. נכניס את העובד החדש במקום המתאים בעץ $O(\log n)$ EmployeesByID
7. נכניס את העובד החדש במקום המתאים בעץ $O(\log n)$ EmployeesBySalary
8. נמצא את העובד המרוויח ביותר ע"י סיור בעץ EmployeesBySalary ונשמור אותו במשתנה recordEarner $O(\log n)$
9. נכניס את העובד החדש במקום המתאים בעץ employeesByID בחברה המתאימה $O(\log n)$
10. נכניס את העובד החדש במקום המתאים בעץ employeesBySalary בחברה המתאימה $O(\log n)$
11. נבדוק האם המשכורת של העובד החדש היא הגבוהה ביותר בחברה, ואם כן נעדכן בהתאם. $O(\log n)$.

סה"כ – סיבוכיות זמן : $O(\log n + \log k)$

RemoveEmployee - בעת הסרת עובד נבצע את השלבים הבאים:

1. נחפש את המפתח EmployeesID בעץ ,EmployeesByID אם קיים נזרוק חריגה $O(\log n)$
2. לאחר שמצאנו, נגש לחברה בה הוא עובד (דרך המצביע) ונחפש אותו בעץ companyEmployeesID/Salaryלפי המשכורת/המזהה הידועים לנו ונמחק אותו $O(\log n)$
3. באותו אופן, נמחק אותו גם מהעץ $O(\log n)$. EmployeesBySalary
4. נמצא את העובד המרוויח ביותר בכל המערכת ע"י סיור בעץ EmployeesBySalary ונשמור אותו במשתנה $O(\log n)$ recordEarner
5. נבצע חיפוש נוסף בעץ employeesBySalary של החברה אחר העובד המרוויח ביותר אחריו ונעדכן בהתאם בשדה של החברה $O(\log n)$
6. נבדוק האם החברה ריקה מעובדים, אם כן נמחק אותה מהעץ $O(\log n)$,NonEmptyCompanies
7. נמחוק את העובד מהעץ של כל העובדים $O(\log n)$

סה"כ – סיבוכיות זמן : $O(\log n)$

בסעיף 6 הסיבוכיות היא לכל היותר $O(\log n)$ כפי שהערנו בעמוד 1

RemoveCompany - בעת הסרת חברה נבצע את השלבים הבאים :

1. נחפש בעץ החברות לפי תעודת הזהות של החברה, אם לא קיימת נזרוק חריגה$O(\log k)$
2. נוודא שאין לחברה עובדים (אם אחד מעצי העובדים ריק) , אם יש נזרוק חריגה$O(1)$
3. נמחוק את החברה מעץ החברות$O(\log k)$

סה"כ – סיבוכיות זמן :$O(\log k)$

GetCompanyInfo - נבצע את השלבים הבאים:

1. נחפש בעץ החברות לפי תעודת הזהות של החברה, אם לא קיימת נזרוק חריגה$O(\log k)$
2. נחזיר את שווי החברה ומספר העובדים לפרמטרים הנתונים$O(1)$

סה"כ – סיבוכיות זמן :$O(\log k)$

GetEmployeeInfo - נבצע את השלבים הבאים:

1. נחפש בעץ העובדים לפי תעודת הזהות של העובד, אם לא קיים נזרוק חריגה$O(\log n)$
2. נחזיר את המשכורת , הדרגה ואת מזהה המעסיק לפרמטרים הנתונים$O(1)$

סה"כ סיבוכיות זמן :$O(\log n)$

IncreaseCompanyValue - בעת העלאת שווי החברה, נבצע את השלבים הבאים:

1. נחפש בעץ החברות לפי תעודת הזהות של החברה, אם לא קיימת נזרוק חריגה$O(\log k)$
2. נוסיף את הערך הנתון לשווי החברה$O(1)$

סה"כ סיבוכיות זמן :$O(\log k)$

PromoteEmployee - בעת קידום עובד, נבצע את השלבים הבאים :

1. נחפש בעץ העובדים לפי תעודת הזהות של העובד, אם לא קיים נזרוק חריגה$O(\log n)$
2. נסיר אותו מעץ העובדים לפי המשכורת$O(\log n)$
3. נסיר אותו מעץ העובדים לפי המשכורת בחברה המעסיקה שלו$O(\log n)$
4. נוסיף לעובד את המשכורת ונקדם אותו בדרגה אם צריך$O(1)$
5. נחזיר את העובד לשני העצים (עשינו זאת על מנת לעדכן את מיקום העובד לאחר השינוי.$O(\log n)$
6. נעדכן את השדה highestEarner של המערכת ושל החברה בה הוא עובד.$O(\log n)$

סה"כ סיבוכיות זמן :$O(\log n)$

HireEmployee - בעת העברת עובד מחברה לחברה, נבצע את השלבים הבאים :

1. נחפש בעץ העובדים לפי תעודת הזהות של העובד, אם לא קיים נזרוק חריגה$O(\log n)$
2. נחפש האם החברה החדשה קיימת בעץ החברות, אם לא קיימת נזרוק חריגה$O(\log k)$
3. נוודא שהעובד לא מועסק כבר בחברה החדשה, כלומר לבדוק האם החברה אליה הוא מצביע היא newCompanyID . $O(1)$
4. לאחר שמצאנו, נגש לחברה הישנה בה הוא עבד (דרך המצביע) ונחפש אותו בעץ companyEmployeesByID/Salaryלפי המשכורת/המזהה הידועים לנו ונמחק אותו $O(\log n)$

5. נבדוק האם החברה הישנה ריקה מעובדים, אם כן נמחק אותה מהעץ NonEmptyCompanies, $O(\log n)$
6. אם החברה הישנה לא ריקה נבצע חיפוש בעץ companyEmployeesBySalary של החברה הישנה אחר העובד המרוויח ביותר ונעדכן בהתאם בשדה של החברה הישנה$O(\log n)$
7. נבדוק האם החברה החדשה ריקה, אם כן נוסיף אותה ל$O(\log n)$, NonEmptyCompanies
8. נכניס את העובד לעץ companyEmployeesByID/Salary של החברה החדשה$O(\log n)$
9. נבצע חיפוש בעץ companyEmployeesBySalary של החברה החדשה אחר העובד המרוויח ביותר ונעדכן בהתאם בשדה של החברה החדשה$O(\log n)$
10. נעדכן את שדה המעסיק של העובד להיות החברה החדשה.$O(1)$

סה"כ *סיבוכיות הזמן היא*$O(\log n + \log k)$

AcquireCompany - כאשר חברה א' רוכשת את חברה ב' נבצע את הפעולות הבאות:

1. נחפש בעץ החברות לפי תעודת הזהות של החברה הרוכשת, אם לא קיימת נזרוק חריגה$O(\log k)$
2. נחפש בעץ החברות לפי תעודת הזהות של החברה הנרכשת, אם לא קיימת נזרוק חריגה$O(\log k)$
3. נוודא ששווי החברה הרוכשת הוא פי 10 מהחברה הנרכשת$O(1)$
4. כעת, שתי החברות בידנו והרכישה יכולה להתבצע. נרצה למזג את שני עצי העובדים (פעם אחת עבור העץ של המשכורות ופעם אחת עבור העץ של המזהים) של כל חברה לעץ אחד הכולל את כולם השומר על המיון לפי המשכורת:
   4.1. נבצע סיור inorder על עץ א' ונשמור במערך בגודל $n_{acquirerID}$
   4.2. נבצע סיור inorder על עץ ב' ונשמור במערך בגודל$n_{targetID}$
   4.3. נבצע מיזוג מערכים ממויינים לתוך מערך בגודל$n_{acquirerID} + n_{targetID}$
   4.4. נעדכן בהתאם את הפרטים של החברה המעסיקה אצל כל עובד
   4.5. נבצע אלגוריתם רקורסיבי הבונה עץ מאוזן בהינתן מערך ממוין סיבוכיות זמן של אלגוריתם זה הוא $O(n_{acquirerID} + n_{targetID})$. *הוכחה בעמוד האחרון*
5. *נשמור את העצים הממוזגים תחת החברה הרוכשת*$O(1)$
6. *נמחק את החברה הנרכשת מהעץ*$O(\log k)$, NonEmptyCompanies
7. *נמחק את עצי ב' (משכורות ומזהים) ונשלח את החברה הנרכשת לפונקציית המחיקה*$O(\log k)$
8. *נחשב את השווי החדש של החברה לפי הדרישות*$O(1)$
9. *נעדכן את העובד המרוויח ביותר בחברה הרוכשת*$O(n_{acquirerID} + n_{target ID})$

סה"כ סיבוכיות הזמן היא$O(\log k + n_{acquirerID} + n_{targetID})$
נשים לב שבסעיף 6, בעץ NonEmptyCompanies יש לכל היותר k איברים

GetHighestEarner - נבצע את השלבים הבאים:

1. $companyID > 0$:
   1.1. נחפש את החברה לפי תעודת הזהות שלה בעץ החברות, אם לא קיימת חברה כזו נזרוק חריגה$O(\log k)$
   1.2. נחזיר את תעודת הזהות של העובד ששמור בטיפוס Company תחת השדה highestEarner, נדאג מראש שהוא יהיה בעל התעודת זהות הנמוכה במקרה של שוויון בשכר (ע"י אופרטור השוואה שנגדיר) אם החברה ריקה נזרוק שגיאה.$O(1)$
2. אם: $companyID < 0$
   2.1. נחזיר את תעודת הזהות של העובד ששמור לנו במשתנה recordEarner באופן דומה, נדאג שיהיה בעל המזהה הקטן ביותר במקרה של שוויון. אם אין עובדים במערכת נזרוק שגיאה$O(1)$

סה"כ סיבוכיות זמן : $O(\log k)$ או $O(1)$ כנדרש.

GetAllEmployeesBySalary - נבצע את השלבים הבאים:

1. $companyID > 0$:
   1.1. נחפש את החברה לפי המזהה שלה בעץ החברות. אם לא קיימת חברה כזו נזרוק חריגה $O(\log k)$
   1.2. נקצה מערך בשם array בגודל $n_{companyID}$ (בעזרת ה countשל אחד מעצי העובדים בחברה) אם אין עובדים בחברה נזרוק שגיאה $O(1)$
   1.3. נקצה מערך בשם elems בגודל $n_{companyID}$ שיחזיק את המצביעים לעובדים לפי הסדר $O(1)$
   1.4. נבצע סיור inOrder על העץ EmployeesBySalary של החברה (זה עובר על העובדים בסדר <u>עולה</u>) ונשמור במערך elems את המצביעים של העובדים $O(n_{companyID})$
   1.5. מכיוון שהתבקשנו לשמור בסדר יורד, נעבור הפוך על elems ונכניס למערך Employees את המזהה של כל עובד.$O(n_{companyID})$
   1.6. נגדיר$O(1)$ $NumOfEmployees = n_{companyID}$
   1.7. נבצע השמה של המערך array לתוך הפרמטרEmployees
2. אם: $companyID < 0$
   2.1. נקצה מערך בגודל n (נקבל את הגודל מהשדה שבעץ EmployeesByID למשל $O(1)$)
   2.2. אם אין עובדים מערכת נזרוק חריגה $O(1)$
   2.3. נקצה מערך בשם elems בגודל n שיחזיק את המצביעים לעובדים לפי הסדר $O(1)$
   2.4. נבצע סיור inOrder על העץ) EmployeesBySalary זה עובר על כל העובדים בסדר <u>עולה</u>) ונשמור במערך את המזהים של העובדים $O(n)$
   2.5. מכיוון שהתבקשנו לשמור בסדר יורד, נעבור הפוך על elems ונכניס למערך Employees את המזהה של כל עובד$O(n)$
   2.6. נגדיר$O(1)$ $NumOfEmployees = n$
   2.7. נבצע השמה של המערך array לתוך הפרמטר $O(1)$ employees

סה"כ סיבוכיות זמן : $O(n_{companyID})$ או $O(n)$ כנדרש.

GetHighestEarnerInEachCompany - נבצע את השלבים הבאים:

1. נבדוק האם $NonEmptyCompanies \rightarrow getSize() < NumOfCompanies$, אם כן נזרוק חריגה. מתבצע ב $O(1)$*כי יש לנו שדה של גודל בעץ.*
2. נקצה מערך לתוך הפרמטר $Employees$ בגודל$O(1)$ $NumOfCompanies$
3. נקצה מערך בשם $Comps$ בגודל $NumOfCompanies$ שיחזיק את החברות שמהם נוציא את העובדים המרוויחים ביותר.
4. נגדיר$O(1)$ $c = 0$
5. נתחיל סיור inOrder על העץ ,NonEmptyCompanies כל פעם שנעצור על חברה נכניס אותה למערך ,Comps ונקדם אתc.
   נעצור את הסיור כאשר $c = NumOfCompanies$. *כלומר המערך הגיע לגודלו הרצוי.*
   $O(\log k + NumOfCompanies)$ הסבר לסיבוכיות זו למטה.
6. נעבור על המערך Comps ועבור כל חברה בו נוציא מהשדה HighestEarner שלה את העובד המרוויח ביותר ונשים את ה idשלו במערך$O(NumOfCompanies)$ Employees

הסבר:

כאשר עושים סיור inOrder השלב הראשון הוא להגיע לתחתית העץ דרך הענף השמאלי, וזה $O(\log k)$. לאחר מכן בוודאות נבצע רק עוד $NumOfCompanies$ פעולות מכיוון שבהכרח החברות לא ריקות כלומר כל צומת בעץ רלוונטית ומקדמת את.c

GetNumEmployeesMatching - נבצע את השלבים הבאים:

1. אם: $companyID > 0$
   1.1. נבצע חיפוש בעץ companies ונבדוק האם קיימת חברה כזו, אחרת נזרוק חריגה$O(\log k)$
   1.2. נבדוק האם החברה שמצאנו ריקה, אם כן נזרוק חריגה$O(1)$
   1.3. נבצע חיפוש בעץ companyEmployeesByID לפי MinEmployeeID ונמצא (אם קיים) את העובד הראשון בטווח. נבצע מעין סיור inorder החל מהעובד הנ"ל במעלה המזהים. עבור כל עובד בטווח נבדוק האם המשכורת שלו ודרגתו מתאימים , ואם כן נסכום לתוך NumOfEmployees.נעצור כאשר הגענו לסוף העץ או שהגענו לעובד שהמזהה שלו גדול יותר מ- MaxEmployeeId.
   סיבוכיות זמן של פעולה זו היא $O(\log n_{companyID} + TotalNumOfEmployees)$: בדומה להסבר בעמוד הקודם.
2. אם $companyID < 0$: נבצע את אותה הפעולה ב1.2. רק על העץ EmployeesByID. אך כעת הסיבוכיות תלויה בכמות כל העובדים במערכת $O(\log n + TotalNumOfEmployees)$.

במקרה ואין עובדים בטווח (בכל אחד מן המקרים) נחזיר אפס .

סה"כ סיבוכיות הזמן היא : $O(\log k + \log n_{companyID} + TotalNumOfEmployees$ *או* $O(\log n + TotalNumOfEmployees)$ *כנדרש.*

Quit *- נבצע את השלבים הבאים:*

1. *סיור inOrder על העץ* EmployeesByID *והפעלת delete על כל החול* $O(n)$
2. *סיור inOrder על העץ* EmployeesBySalary *והפעלת delete על כל הצמתים* $O(n)$
3. *סיור inOrder על העץ* NonEmptyCompanies *ועבור כל חברה נסייר בשני תתי העצים שלה* CompanyEmployeesByID *ו-* CompanyEmployeesBySalary *והפעלת delete על כל הצמתים (בסיום הסיור בתתי העצים שלה)-* $O(k + 2n)$ *, כי סך הכל בכל התתי עצים נבצע* $2n$ *פעולות, כלומר*$O(n + k)$
4. *סיור inOrder על העץ* Companies *ו* NonEmptyCompanies*והפעלת delete על כל הצמתים, אין צורך לסייר בתתי העצים של כל חברה, מכיוון שניגשנו אליהם דרך המצביעים ששמרנו ב,* NonEmptyCompanies*לכן סך הכל*$O(k)$
5. *סך הכל הסיבוכיות היא :*$O(n + k)$

סיבוכיות מקום של כל המערכת:

מבני הנתונים במערכת שמוקצים דינאמית:

- Companies: - עץ המכיל k צמתים ככמות החברות במערכת $O(k)$
- NonEmptyCompanies - עץ המכיל מקסימום k צמתים ככמות החברות במערכת $O(k)$
- EmployeesByID: - עץ המכיל n צמתים ככמות העובדים במערכת$O(n)$
- EmployeesBySalary - עץ המכיל n צמתים ככמות העובדים במערכת $O(n)$
- בטיפוס של Company יש שדה employeesByID המכיל את כל עובדי החברה. מכיוון שכל עובד מועסק רק בחברה אחת אז סך כל הצמתים המוחזקים בעצים האלה הוא$O(n)$ n
- בטיפוס של Company יש עץ employeesBySalary המכיל את כל עובדי החברה. מכיוון שכל עובד מועסק רק בחברה אחת אז סך כל הצמתים המוחזקים בעצים האלה הוא$O(n)$n
- recordEarner: - מצביע לעובד המרוויח ביותר בכל המערכת.$O(1)$

סך הכל נקבל כי סיבוכיות המקום במערכת היא$O(n + k)$

<u>הסבר לסיבוכיות של האלגוריתם שבונה עץ AVL ממערך ממויין</u>
(עמוד 4 – AcquireCompany)

האלגוריתם קובע את מרכז המערך ליהיות השורש ואז שולח לריקורסיה אחת את חצי המערך הימני ולריקורסיה נפרדת את החצי השמאלי, לכן אם $T(n)$ הוא סיבוכיות הזמן אז מתקיים :

$$T(n) = 2T\left(\frac{n}{2}\right) + C$$

C הוא הזמן הקבוע שלוקח לנו לחשב את אמצע המערך ולחבר את השורש לתת העץ השמאלי ולתת העץ הימני.
לפיכך אם נחשב סיבוכיות בעזרת הצבה חוזרת נקבל :

$$T(n) = 2^{\log_2 n} \cdot T(0) + C = n + C$$

לכן הסיבוכיות היא $O(n)$ כאשר n הוא גודל המערך כנדרש.