**עבודת סיום - קורס קיץ C#**

**הנדסאים אריאל - תשע"ט**

**שם הסטודנט:** אלמוג עמוס.

**ת.ז:** 305167397.

**שם הסטודנט:** גיא בלוך.

**ת.ז:** 305292534.

**שם הסטודנט:** ירין ענבר.

**ת.ז:** 205927536.

.C**קורס:** #

**שם המרצה:** מר אייברס דני.

**תוכן עניינים**

.................................................................................................3BinaryTree

....................................................................................................10ArrayList

..................................................................................................14Dictionary

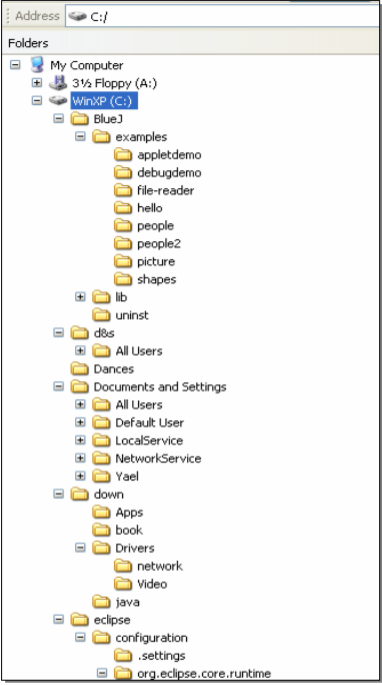
**Binary Tree – עץ בינארי**

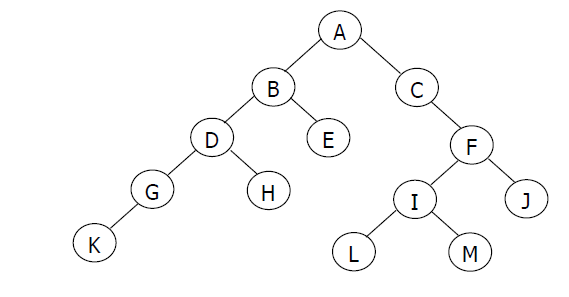
עץ בינארי הוא מבנה נתונים המורכב מצמתים וקשתות ומקיים את התכונות הבאות:

1. בראש העץ נמצא צומת השורש, שממנו מסתעף העץ כלפי מטה.
2. .מכל צומת יוצאות, לכל היותר, שתי קשתות, קשת יחידה או אף לא קשת אחת.
3. כל צומת, פרט לשורש, נמצא בקצה התחתון של קשת אחת בלבד.

לדוגמא, באיור הבא ניתן לראות חלק ממערכת קבצים ששורשה הוא “My Computer”.

בכל ספריה במערכת הקבצים, יש ספריות או קבצים נוספים.



בעץ הבינארי מתקיימת "מערכת יחסים" בין הצמתים בדומה ליחסים בעץ המשפחתי, כמו יחסי אב -בן, יחסי אחים וכו'. נגדיר את המונחים בעזרת העץ הבינארי הבא :

* A הוא אב (FATHER) של B ו-C.
* B הוא בן שמאלי (Left Son) של A .
* C הוא בן ימני (Right Son) של A.
* אחים(Brothers) של שני צמתים שהם בנים לאותו אב.
* עלה(Leaf) – צומת שאין לו בנים.
* צומת פנימי (Internal Node) צומת שיש לו לפחות בן אחד.
* אב קדמון (Ancestor) הוא אב או אבי האב או אביו של אבי האב וכן הלאה.
* צאצא (desecendant) הוא בן או בנו של הבן או נכדו של הבן וכן הלאה.

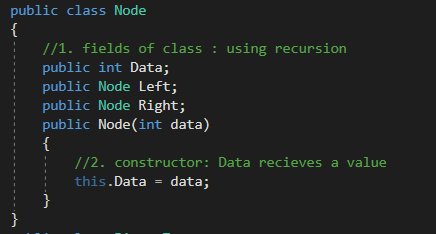
ניתן לחלק את העץ לרמות כך שכל צומת בעץ שייך לרמה מסוימת.

* רמה(Level)/ דרגה (Rank) של צומת בעץ בינארי מוגדר כך :
  1. רמת השורש של העץ הוא 0.
  2. רמת כל צומת אחר בעץ, גדולה באחד מהרמה של האב שלו.
* עומק העץ ( Tree Depth)/ רמת העץ (Tree Level) הרמה הגבוהה ביותר בעץ.

מהלך התכנית :

1) יצירת מחלקה בשם Node שכוללת בתוכה את השורש (Data) ועוד 2 מצביעים מסוג Node (המחלקה) עבור עץ ימין ועבור עץ שמאל.

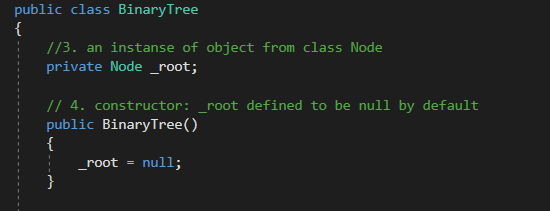
2) הגדרת הבנאי של המחלקה כך שארגומנט שנשלח אליה יוצב במשתנה DATA.



3) יצירת מחלקה חדשה בשם Binary Tree , שמכילה משתנה, ששמו \_root , שהוא בעצם מחלקה מסוג Node .

4) הגדרת בנאי. הבנאי קובע שבכל יצירה חדשה של עץ, ה \_root שלו יאותחל בnull.

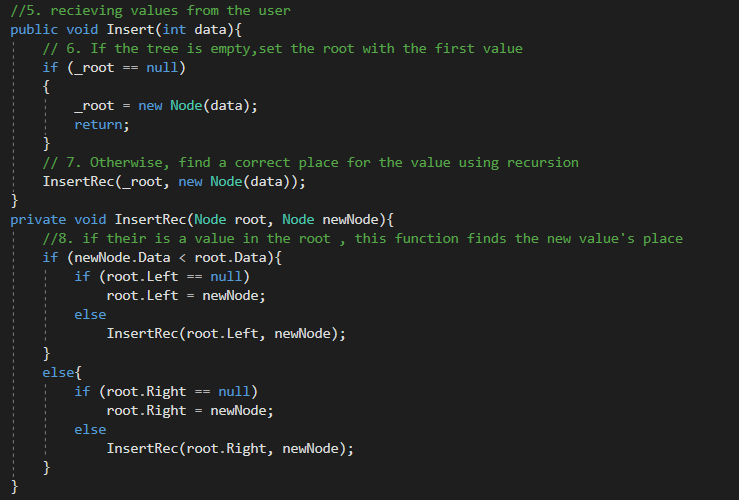
5) כתיבת פונקציה לקליטת נתונים לעץ מהמשתמש.



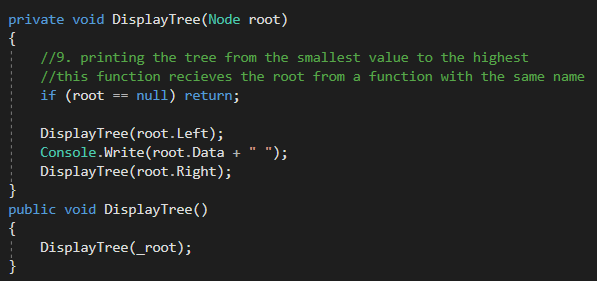
6) אם אין עדיין נתונים בעץ(כלומר זה עץ חדש) אז הנתון הראשון שהוכנס יהיה ה- \_root שלנו .

7) אם כבר יש לנו נתון בעץ (ה- \_root כבר קיים) אז נקרא לפונקציה אחרת (InsertRec) שתדע לסדר לנו את הנתון החדש לפי מיקומו ההגיוני בעץ.

8) הפונקציה InsertRec בודקת אם הנתון החדש שהמשתמש הכניס גדול מה-\_root או קטן ממנו. אם הוא קטן , והמחלקה left של ה-\_root פנויה אז היא מכניסה לשם את הנתון החדש. אם המחלקה left לא פנויה, אז הפונקציה קוראת לעצמה מחדש בצורה רקורסיבית, כשהיא שולחת את root.left והנתון החדש בתור ארגומנטים , עד שמוצאת מקום ריק להכניס את הנתון החדש.



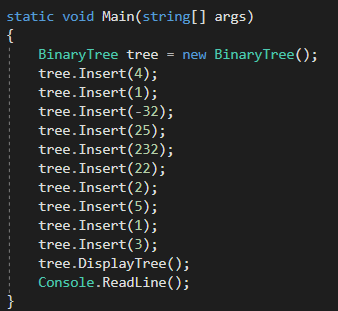
אם הנתון גדול מה- \_root הפונקציה בודקת את root.right וחוזרת על התהליך הנ"ל עד שמוצאת מקום ריק להכניס את הנתון החדש.



9) DisplayTree היא פונקציה להדפסת העץ מהנתון הקטן ביותר עד לנתון הגדול ביותר.   
זוהי פונקציה רקורסיבית . לכן בהתחלה נראה בדיקה שלה אם ה-root ריק . אם הוא ריק היא חוזרת אם לא, היא קוראת לעצמה עם root.left (תחילה עם הערכים הקטנים יותר) עד שהיא מגיעה ל"מבוי סתום" , כלומר לרכיב בעץ שאין לו יותר ערכים בענפים השמאליים, ואז היא מדפיסה את אותו רכיב, ובודקת שוב בצורה רקורסיבית אם יש לאותו רכיב ערכים בענפים הימניים שלו וכן הלאה.

לאחר שסיימה עם כל הערכים הללו, הפונקציה חוזרת להדפסה של \_root וקוראת לעצה בצורה רקורסיבית , אך הפעם שולחת את הענפים שמימינה , הערכים שגדולים ממנה, וכך מדפיסה אותם עד שמסיימת עם כל העץ.

כעת לבסוף יש לייצר אובייקט חדש מתוך מקלאס BinaryTree, לאחר מכן יש לקרוא לאובייקט ולהשתמש בפונקציה ה insert על מנת להכניס ערכים חדשים,  
כעת נשתמש בפונקציית ה DisplayTree על מנת להציג את הערכים.



הפלט שהתקבל:



**מטלת כיתה – עץ בינארי**

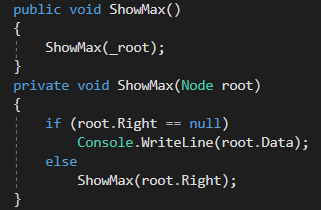
כתוב פונקציה המדפיסה למסך את הערך הכי גדול בעץ.

**פתרון למטלת כיתה – עץ בינארי**

נכתוב פונקציה בשם ShowMax שלא מקבלת ארגומנטים, שהמשתמש יוכל לקרוא לה.

פונקציה זו תקרא לפונקציה אחרת בעלת אותו שם שכן מקבלת ארגומנטים, ונשלח אליה את שורש העץ כארגומנט.

נבצע בדיקה אם קיים ערך בענף הימני של העץ. אם לא קיים ערך, כלומר \_root.right = null, זה אומר לנו ששורש העץ הוא גם האיבר הכי גדול שלנו ולכן נדפיס אותו למסך. אם אכן קיים ערך בענף הימני, נקרא שוב לפונקציה, בצורה רקורסבית, אך הפעם נשלח אליה את הענף הימני של ­\_root כארגומנט.



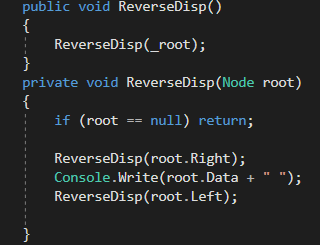
**מטלת בית – עץ בינארי**

כתוב פונקציה המדפיסה את העץ הפוך – כלומר מהערך הכי גדול לערך הכי קטן.

**פתרון למטלת הבית – עץ בינארי**

נכתוב פונקציה שלא מקבלת ארגומנטים בשם ReverseDisp , שהמשתמש יוכל לקרוא לה. פונקציה זו תקרא לפונקציה בעלת אותו שם שכן מקבלת ארגומנטים, ונשלח לה את שורש העץ (\_root) .

מהלך העבודה של הפונקציה הזו היא בדיקה עם הערך שהיא מקבלת הוא ריק (null) . אם כן היא חוזרת. אם לא היא קוראת לעצמה שוב בצורה רקורסיבית, אך הפעם היא שולחת כארגומנט את הענף הימני. לאחר שהיא הגיעה לענף הימני האחרון, היא מדפיסה אותו למסך ובודקת אם קיים לו ענף שמאלי. אם קיים לו ענף שמאלי היא תבדוק אם לאותו ענף יש עוד בנים ימניים. אם לא תדפיס את אותו ענף שמאלי ותחזור בחזרה עד שתגיע לבסוף ל-\_root. היא תדפיס את ה-\_root ואז תבדוק את הבנים השמאליים שלו וכן הלאה וכך למעשה היא מדפיסה את כל העץ בסדר הפוך.



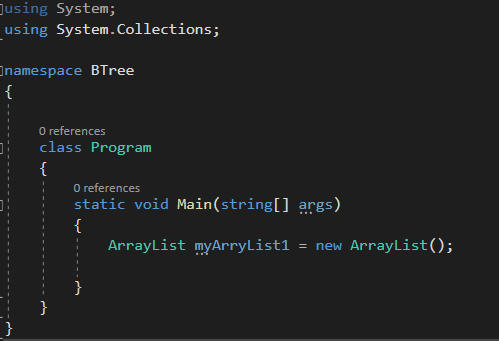
**ArrayList**

ה ArrayList היא מחלקה לא גנרית (כלומר יכולה להחזיק מספר אלמנטים מסוגים שונים) בשפת C#, ה ArrayList יכול להחזיק בתוכו מספר אלמנטים מסוגים שונים כמו int, byte, bool, string ועוד.

ב ArrayList אין צורך לציין את כמות האיברים במערך, הוא יודע לגדול ולקטון בצורה אוטומטית.   
לפי שנתחיל יש להוסיף בתחילת הקוד ( לפני ה main):

using System.Collections;

ה using הזה עוזר לנו לממש את המחלקה הבנויה מראש ArrayList, כעת נוכל לממש אותה בצורה הפשוטה ביותר ב c# כמו שממשנו משתנים מסוגים שונים(int, char וכו'):



כך, בצורה הפשוטה ביותר יצרנו מערך דינמי וקראנו לו myArrayList1, המערך אינו מוגבל רק לסוג אחד של משתנים או מוגבל בגודלו.

ל ArrayList יש מספר מתודות הגדרה הבנויות מראש הם מפורטות בטבלה מטה:

| הגדרות | תיאור |
| --- | --- |
| Capacity | מגדיר או מקבל את מספר הערכים שה ArrayList יכול להכיל. |
| Count | מציג את המספר האמתי של האיברים במאוכסנים במערך |
| IsFixedSize | מקבל ערך בוליאני אשר קובע אם המערך הוא בגודל קבוע או לא. |
| IsReadOnly | מקבל ערך בוליאני אשר קובע אם ניתן לבצע אך ורק קריאה של איברי המערך ללא אפשרות לשינוי |

| מתודות | תיאור |
| --- | --- |
| [Add()/AddRange()](https://www.tutorialsteacher.com/csharp/csharp-arraylist#add) | המתודה Add מוסיפה איבר בודד לסוף המערך.  המתודה AddRange מוסיפה מספר איברים למערך (לסופו). |
| [Insert()/InsertRange()](https://www.tutorialsteacher.com/csharp/csharp-arraylist#insert) | הפונקציה insert מוסיפה איבר למערך על פי האינדקס שצויין.  הפונקציה InsertRange מוסיפה מספר איברים למערך, היא מקבלת את האינדקס שממנו יש לבצע הכנסה וגם את המערך שיש להוסיף.  כלומר במערך x יש: 1,2,5 ואנו רוצים להוסיף אותו לתוך מערך y אם נשתמש ב insertRange(0,x) הוא יוסיף את המערך x לתוך המערך y במיקום ה-0. |
| [Remove()/RemoveRange()](https://www.tutorialsteacher.com/csharp/csharp-arraylist#remove) | המתודה Remove מסירה איבר במערך על מי מיקומו באינדקס,  המתודה RemoveRange מסירה מספר איברים במערך על פי האינדקס שצוין, הפונקציה מקבלת את האינדקס הראשון שבו היא מתחילה ולאחר מכן את כמות האיברים למחיקה.  אם לדוגמא במערך: 1,4,”rer”,false  נפעיל רק את remove(1) בעצם מחקנו את הערך false מתוך המערך,  אם נפעיל את RemoveRange(0,2) הפונקציה תתחיל למחוק האיבר במקום ה-0 ותתקדם עוד מקום אחד (כיוון שהיה עליה למחוק את האיבר במקום 0,אז היא תתקדם ותמחק גם את האיבר הבא אחריו) |
| [RemoveAt()](https://www.tutorialsteacher.com/csharp/csharp-arraylist#removeat) | המתודה RemoveAt מסירה איבר אחד לפי מיקומו באינדקס. |
| [Sort()](https://www.tutorialsteacher.com/csharp/csharp-arraylist#sort) | מסדר את המערך, רק אם האיברים הם מאותו סוג. |
| [Reverse()](https://www.tutorialsteacher.com/csharp/csharp-arraylist#sort) | הופך את איברי המערך – אם המערך המקורי: 2,3,7,4 אז אחרי הפעלת המתודה : 4,7,3,2 |
| [Contains](https://www.tutorialsteacher.com/csharp/csharp-arraylist#contains) | מחזיר ערך בוליאני אשר מייצג אם האיבר שהוכנס אליו קיים או לא.  לדוגמא: “re”,4,7,true,443,”1”,2  אם נפעיל את הפונקציה contains ונשלח לה את המספר 1 אנו נקבל false מכייון ש-1 הוכנס למערך בתור string ולא בתור int. |
| Clear | מסיר את כל איברי המערך |
| CopyTo | מעתיק את כל איברי המערך ל ArrayList חדש (יש לציין לאיזה ArrayList יש להעביר) |
| IndexOf | מחזיר את המיקום של הערך אם קיים במערך, אם לא מחזיר -1 |

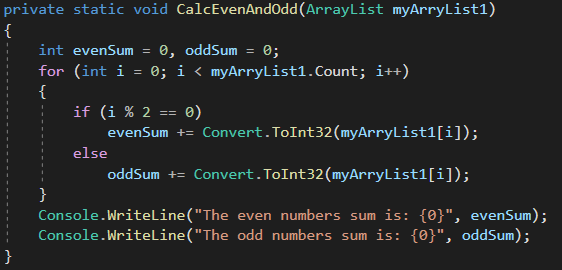
**מטלת כיתה – ArrayList**

כתוב פונקציה הקולטת מערך של ArrayList ומחשבת את כל האיברים במקומות שלמערך יוכנסו מספרים בלבד (חיובים ושליליים).

**פתרון למטלת כיתה – ArrayList**

נכתוב פונקציה בשם CalcEvenAndOdd המקבלת כפרמטר את ה ArrayList ולא מחזירה ערך,   
כעת ניצור 2 משתנים בתוך הפונקציה אחד עבור המקומות הזוגיים הזוגיים ומשתנה נוסף עבור האי-זוגיים.  
נרוץ בלולאת פור על איברי המערך כאשר I מתחיל ב-0 ורץ כל עוד הוא לא הגיע לאיבר האחרון ברשימה.

ונבדוק את מיקום ה-I אם מיקום ה-i זוגי על ידי if(i%2==0) אז הוא יכניס את הערך שנמצא ברשימה לתוך evenSum ואם הוא אי-זוגיים הוא ייכנס לתוך oddSum



**מטלת בית – ArrayList**

כתוב פונקציה הקולטת ערכים מסוג של int, ללא הגבלה על הכמות,  
ולבסוף תאכסן אותם בתוך ArrayList,

הנח שבמערך יש יותר מ-4 איברים.  
יש לבצע חישוב של חיבור האיבר הראשון עם האיבר האחרון, חילוק בין האיבר האחרון לשני ( במקום ה-1) ולהציג את התוצאה של שני החישובים.

**פתרון מטלת הבית – ArrayList**

נכתוב פונקציה בשם calc שמקבלת את ה arrayList כפרמטר, הפונקציה לא מחזירה ערך ולכן נגדיר אותה כ void.

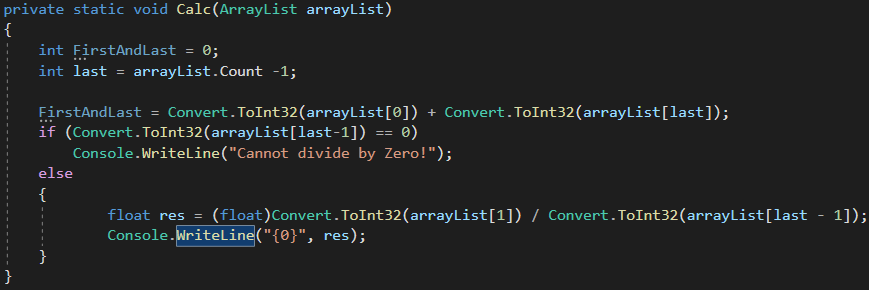
נגדיר משתנה מסוג int המייצג את האיבר האחרון ברשימה, נעשה זאת על ידי השימוש ב property – count, נרשום זאת בתור arrayList.Count -1   
(arrayList זה השם של הפרמטר שקבלנו).

כעת נבצע חישוב פשוט של האיבר במקום ה-0 שהוא הראשון שלנו, ולאחר מכן חיבור לאיבר במקום האחרון.

חשוב לבצע זאת על ידי Convert ללא שימוש במחלקה הזאת – לא נוכל להבטיח שאכן נקבל מספרים.

כעת נבדוק את האיבר אחד לפני האחרון שהוא לא 0 , אם הוא 0 נדפיס הודעה למסך.

אם לא אז נשמור את התוצאה הסופית לתוך float ונציג אותו למסך.



**Dictionary**

בדומה למילון שכולנו מכירים, בין אם בשפה העברית ובין אם באנגלית, אשר מכיל בתוכו אוסף של מילים והגדרותיהם, המופיעות לרוב על פי סדר א"ב באותה השפה – כך גם ה'מילון' (Dictionary) בשפת C#.

ה Dictionary בשפת C# מכיל בתוכו אוסף של מפתחות (Keys) וערכים (Values), כאשר בהשוואה למילון, המפתח (Key) הוא המקביל למילה, והערך (Value) מקביל להגדרת המילה.

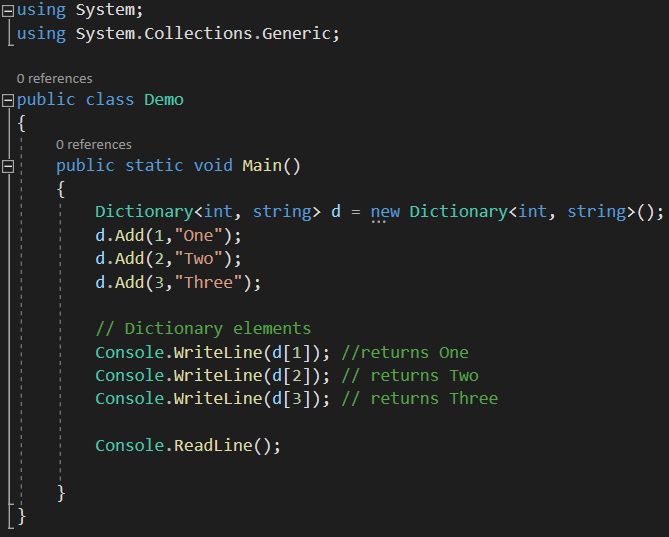
בדוגמא הבאה, ניתן לראות את הגדרת סוג המפתח והערך, תוך הכרזה על אובייקט מסוג מילון (Dictionary).



במקרה הנ"ל, ה Key הוא מסוג Int, וה Value מסוג String. שניהם יאוחסנו בתור אובייקט מסוג Dictionary.

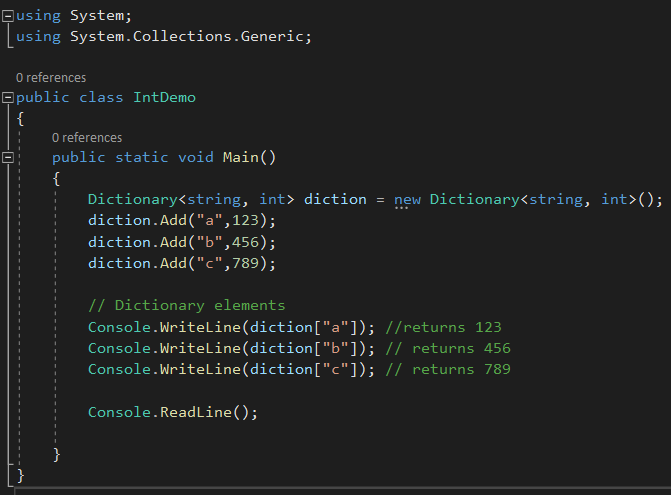
בדוגמא הבאה, ניתן לראות תכנית בה מוגדרים שלושה ערכים מסוג String לתוך Dictionary, ושלושתם מודפסים למסך בזה אחר זה.

הפלט שיודפס למשתמש:



בדוגמא השניה, ניתן לראות תכנית בה מוגדרים שלושה ערכים, הפעם מסוג Int, עם Key’s מסוג String, ושלושתם מודפסים למסך בזה אחר זה.

 הפלט שיודפס למסך:

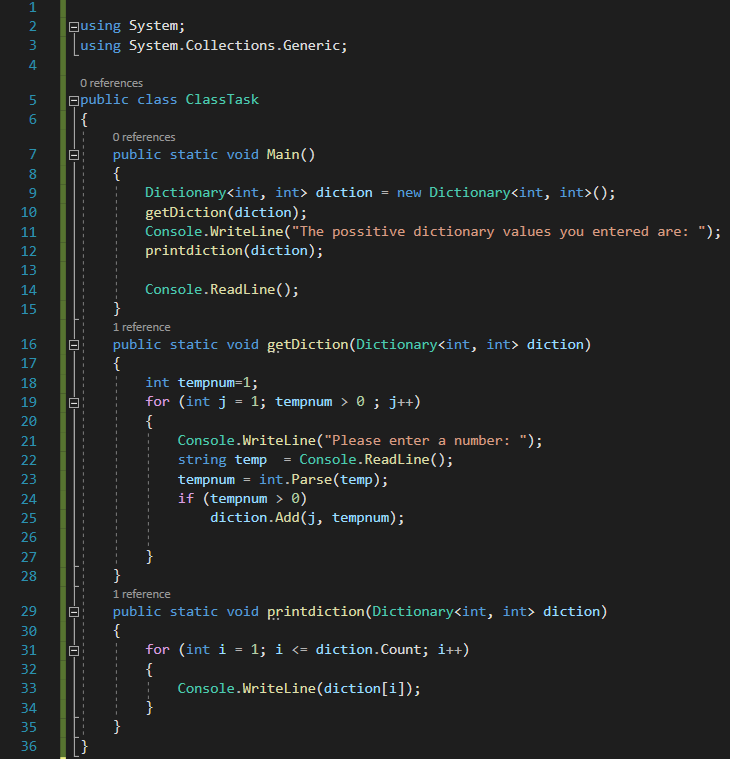


**מטלת כיתה – Dictionary**

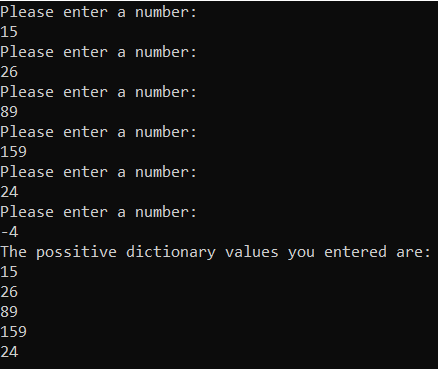
צור תוכנית ובה פונקציה הקולטת מהמשתמש מספרים חיוביים בזה אחר זה (עד אשר נקלט מספר שלילי), וצור פונקציה שמכניסה את אותם המספרים לתוך ערכי Dictionary ע"פ הסדר בו הם נקלטו מהמשתמש.

הדפס למסך את הערכים החיוביים שנקלטו בערכי ה Dictionary .

**פתרון למטלת הכיתה – Dictionary**



לדוגמא, אם נקלטו מהמשתמש המספרים 15, 26, 89, 159, 24, -4, יוצג למסך הפלט הבא:



**מטלת בית – Dictionary**

צור תוכנית הקולטת מהמשתמש 10 מספרים (הנח שהם שלמים).

צור פונקציה שתקלוט את המספרים לתוך מערך ותסדר אותם בסדר עולה.

צור פונקציה שתעביר את הערכים המסודרים לתוך Dictionary.

צור פונקציה שתשאל את המשתמש אם הוא מעוניין שהערכים המסודרים יוצגו למסך, במידה ויבחר Yes, הערכים יודפסו. במידה ויבחר No, יקבל הודעת “Thank you, good bye”.

**פתרון למטלת הבית – Dictionary**

