הרצות לדוגמה:

הרצה 1:

> + Erez Wanderman 123456789 1 10000

New customer create: name = Erez Wanderman id = 123456789 customer number = 1 balance = 10000

> + Elvis Aaron Presley 123444444 2 99999999

New customer create: name = Elvis Aaron Presley id = 123444444 customer number = 2 balance = 99999999

> + David Robert Jones 123555555 3 99995555

New customer create: name = David Robert Jones id = 123555555 customer number = 3 balance = 99995555

> David Robert Jones 3 10000

Customer 3 deposited 10000

> ? MAX

Customers with maximal balance:

Name: David Robert Jones ID: 123555555 Customer Number: 3 Balance: 100005555

End of list.

> Erez Wanderman 1 -10001

Customer 1 deposited -10001

> Elvis Aaron Presley 2 -100000004

Customer 2 deposited -100000004

> David Robert Jones 3 -100005559

Customer 3 deposited -100005559

> ? MINUS

Customers with negative balance:

Name: Erez Wanderman ID: 123456789 Customer Number: 1 Balance: -1

Name: David Robert Jones ID: 123555555 Customer Number: 3 Balance: -4

Name: Elvis Aaron Presley ID: 123444444 Customer Number: 2 Balance: -5

End of list.

> ? MAX

Customers with maximal balance:

Name: Erez Wanderman ID: 123456789 Customer Number: 1 Balance: -1

End of list.

> ? 2

Balance is -5

> Elvis Aaron Presley 2 5

Customer 2 deposited 5

> - 2

Customer number 2 has left the bank.

הרצה 2

> + C1 1 1 -20

New customer create: name = C1 id = 1 customer number = 1 balance = -20

> + C2 2 2 -20

New customer create: name = C2 id = 2 customer number = 2 balance = -20

> + C3 3 3 -40

New customer create: name = C3 id = 3 customer number = 3 balance = -40

> + C4 4 4 -40

New customer create: name = C4 id = 4 customer number = 4 balance = -40

> C1 1 -20

Customer 1 deposited -20

> C3 3 20

Customer 3 deposited 20

> ? MAX

Customers with maximal balance:

Name: C3 ID: 3 Customer Number: 3 Balance: -20

Name: C2 ID: 2 Customer Number: 2 Balance: -20

End of list.

> ? MINUS

Customers with negative balance:

Name: C2 ID: 2 Customer Number: 2 Balance: -20

Name: C3 ID: 3 Customer Number: 3 Balance: -20

Name: C4 ID: 4 Customer Number: 4 Balance: -40

Name: C1 ID: 1 Customer Number: 1 Balance: -40

End of list.

> ? 1

Balance is -40

> ? 2

Balance is -20

> ? 3

Balance is -20

> ? 4

Balance is -40

הסבר התוכנית:

את מבנה הנתונים מנהלת המחלקה BankData. היא מכילה שני עצים אדומים שחורים: customersByNumber ו־customersByBalance. בראשון המפתחות הם מספרי הלקוחות והנתונים הנלווים עם הלקוחות, בשני המפתחות הם היתרות של הלקוחות (ללא כפילויות) והנתונים הנלווים הם כל הלקוחות בעלי יתרה זאת.  
בכל מקרה של פעולה לא חוקית (כגון ניסיון למחוק לקוח ע"פ מספר שלא קיים), נזקרת שגיאה והיא מוצגת למשתמש.  
נסמן את מספר הלקוחות (מספר הצמתים בעץ הראשון) ומספר היתרות הייחודיות (מספר הצמתים בעץ השני) ()

הפעולות שמממשת המחלקה:

* NewCustomer: מוודאים שלא קיים כבר לקוח עם המספר המבוקש ע"י חיפוש המספר בעץ customersByNumber. יוצרים צומת עם מפתח=מספר הלקוח שהתקבל, מידע נלווה=לקוח חדש. מכניסים אותו לעץ customersByNumber.  
  מחפשים בעץ customersByBalance את היתרה שהתקבלה בקלט. אם מוצאים אותה, מוסיפים לרשימה שבמידע הנלווה של הצומת שנמצא את הלקוח. אם לא, יוצרים צומת חדש עם מפתח=היתרה, מידע נלווה=רשימה חדשה, ואז מוסיפים לרשימה את הלקוח ומוסיפים את הצומת לעץ customersByBalance. לבסוף, שומרים על הלקוח שלושה מצביעים: לצומת בעץ customersByBalance, לרשימה המקושרת ולצומת ברשימה.  
  זמן ריצה: חיפוש/הכנסה ל־customersByNumber - , חיפוש/הכנסה ל־customersByNumber - , הכנסה לרשימה . סה"כ זמן ריצה במקרה הגרוע .
* RemoveCustomer: מחפשים בעץ customersByNumber לקוח לפי מספר הלקוח שהתקבל, מוודאים שהוא קיים ושיתרתו אינה 0. בעזרת המצביעים ששמורים ברשומת הלקוח, מוחקים אותו מהרשימה שבצומת של customersByBalance ואם הרשימה התרוקנה אז מוחקים את הצומת מהעץ customersByBalance.  
  מוחקים את הצומת שמצאנו בהתחלה מהעץ customersByNumber.  
  זמן ריצה: חיפוש/מחיקה מהעץ customersByNumber - . מחיקה מרשימה (זו רשימה דו מקושרת). מחיקה מהעץ customersByBalance . סה"כ זמן ריצה במקרה הגרוע הוא .
* DepositOrWithdraw: מחפשים בעץ customersByNumber לקוח לפי מספר הלקוח שהתקבל, מוודאים שהוא קיים והשם שלו זהה לשם שהתקבל. מעדכנים את יתרת הלקוח. מסירים את הלקוח מהעץ customersByBalance באופן דומה למה שתואר בפעולה RemoveCustomer. מוסיפים את הלקוח חזרה לעץ customersByBalance עם היתרה המעודכנת באופן דומה למה שתואר בפעולה NewCustomer.  
  זמן ריצה: חיפוש ב־customersByNumber - . מחיקה/הכנסה לרשימה . מחיקה הכנסה ל־customersByBalance - . סה"כ זמן ריצה במקרה הגרוע הוא .
* GetBalanceByNumber: מחפשים בעץ customersByNumber את הלקוח לפי מספר הלקוח שהתקבל, מוודאים שהוא קיים ומחזירים את השדה balance.  
  זמן ריצה במקרה הגרוע: (חיפוש בעץ customersByNumber)
* GetRichest: מחפשים בעץ customersByBalance את המקסימום ומחזירים את המידע הנלווה - רשימה של הלקוחות בעלי היתרה הגבוהה ביותר. זמן ריצה במקרה הגרוע: (חיפוש בעץ customersByBalance).  
  הערה: אם הפעולה כוללת גם הדפסת כל הלקוחות העונים על התנאי (ולא אחד שרירותי מהרשימה) אז זמן הריצה יהיה במקרה שלכל הלקוחות אותה יתרה.
* GetNegative: יוצרים רשימה חדשה ואז מפעילים את האלגוריתם הרקורסיבי הבא על השורש של העץ customersByBalance:

AddNegativeRecursively(T, x):  
if x=nil[T] then  
 return  
AddNegativeRecursively(T, left[x])  
if key[x]<0 then  
 add each node in the linked list value[x] to our output list  
 AddNegativeRecursively(T, right[x])

האלגוריתם בעצם מבצע סריקה תוכית ומטפל באותו חלק של העץ המכיל ערכים שליליים.  
לבסוף מחזירים את הרשימה.  
זמן ריצה: סריקה של העץ , פעולות הכנסה - סה"כ: במקרה הגרוע - כולם במינוס ☹

הסברים נוספים על התוכנית:

* המחלקות LinkedList, LinkedListNode - רשימה דו מקושרת וצומת ברשימה דו מקושרת.
* המחלקות RBTree, RBTreeNode - עץ אדום שחור, צומת בעץ אדום שחור.  
  במתודת המחיקה בוצע השינוי הבא ביחס לאלגוריתם הנתון בספר:  
  במקרה שבו ל־z יש שני בנים, לא מעתיקים את המפתח והמידע הנלווה של y (העוקב) אליו אלא מעתיקים את המצביעים של z ל־y ומשנים את המצביעים ל־z שיצביעו על y.  
  השינוי גורם לכך שהצומת שנמחק יהיה תמיד הצומת z שנשלח למתודה, וכך לא יישברו המצביעים צומת שיושבים על אובייקטי customer שונים.  
  לשינוי אין השפעה על זמן הריצה האסימפטוטי של DELETE.
* Main.cpp - תוכנית ראשית. בלולאה קוראת את הקלט מהמשתמש, מבצעת את הפקודות ומדפיסה את התוצאות בצורה ידידותית.