רוני נפתלוביץ' 319049060

מטלה מספר 4:

המחלקה tree.tpp

הקובץ tree.tpp מכיל את המימוש של תבנית המחלקה Tree, המייצגת עץ k–ארי כללי שבו כל קודקוד יכול להכיל עד k ילדים. בקובץ הזה גם מימשנו את המחלקה של Node שמייצגת קודקוד יחיד בעץ ומכילה את ערך המפתח שלו, למחלקה הזאת יש 2 שיטות- אחת מאפשרת להוסיף ילד לקודקוד והשנייה מאפשרת לקבל את ערך המפתח של הקודקוד.

שיטות עיקריות:

1. **בנאי** :יוצר עץ חדש, ואנחנו מגדירים בו את סדר העץ שמגדיר את מספר הילדים המרבי של כל קודקוד.
2. **add_root**: השיטה מוסיפה שורש חדש לעץ ומקבלת את ערך המפתח שלו.
3. **add_sub_node**: מוסיף ילד חדש לקודקוד נתון.
4. **renderTree**: מצייר את העץ באמצעות SDL.
5. **iterators**: מספק מספר איטרטורים שונים לסריקת העץ Pre-Order, Post-Order, In-Order, BFS :ו.DFS-
6. **myHeap**: הופך את העץ לערימה בינארית מינימלית.

הספריות שבהן השתמשתי כדי ליצור את העץ ב- GUI: SDL , SDL_ttf-

באמצעות הפקודה SDL_CreateWindow יצרתי חלון חדש שבתוכו יודפס העץ. הפקודה SDL_CreateRenderer יוצרת רנדר חדש שמאפשר לצייר על החלון שיצרנו.

כדי לצייר את העץ השתמשתי בפקודות של הספרייה כמו SDL_RenderDrawLine , SDL_RenderFillRect , SDL_RenderCopy שעזרו לי לייצג כל תא בעץ באמצעות מלבן כחול ולייצר קו שמחבר בין אבא לילדים שלו.

המחלקה complex.hpp

הקובץ complex.hpp מייצג את המחלקה Complex שמייצגת מספר מרוכב. בתור שדות היא מכילה את החלק הממשי והחלק המרוכב של המספר.

שיטות עיקריות:

1. **בנאי** :יוצר מספר מרוכב חדש, מקבל שני ערכים real ו- imag.
2. :**operator<<**מפעיל את האופרטור <<להדפסת מספרים מרוכבים .
3. :**operator==**מפעיל את האופרטור ==להשוואת שני מספרים מרוכבים.
4. :**operator!=**מפעיל את האופרטור =!להשוואת שני מספרים מרוכבים.
5. :**operator>**מפעיל את האופרטור >להשוואת מספרים מרוכבים על בסיס הנורמה שלהם.
6. :**operator<**מפעיל את האופרטור <להשוואת מספרים מרוכבים על בסיס הנורמה שלהם.

המחלקה tree_iterators.tpp

הקובץ tree_iterators.tpp מכיל את המימוש של האיטרטורים השונים למחלקת Tree. האיטרטורים מספקים אפשרות לסרוק את העץ בדרכים.

האיטרטורים שמימשתי:

- **PreOrderIterator**: איטרטור המאפשר סריקה בסדר Pre-Order.
- **PostOrderIterator**: איטרטור המאפשר סריקה בסדר Post-Order.
- **InOrderIterator**: איטרטור המאפשר סריקה בסדר In-Order.
- **BFSIterator**: איטרטור המאפשר סריקה בשיטת BFS .
- **DFSIterator**: איטרטור המאפשר סריקה בשיטת DFS .
- **HeapIterator**: איטרטור ההופך את העץ לערימה בינארית מינימלית ומספק סריקה בהתאם.

המחלקה test.cpp :

המחלקה אני לוקחת דגימות מכמה טיפוסים של עצים חדשים- double, int, string, char, complex.
לכל מקרה אני דוגמת את הפונקציות של המחלקה tree, כמו get_root ו- getValue לאמת החזרת ערך אמיתי.

במקרה של עץ מרוכב בדקתי גם את האופרטורים שמימשתי במחלקה של מספרים מרוכבים ובדקתי שהתוצאות הן התוצאות המצופות בכל אופרטור.
בעץ של מחרוזת בדקתי גם את הדפסת העץ ב-GUI.

ובמחלקה של דאבל בדקתי נסיון של הוספת כמות ילדים לשורש מעבר למספר K שהגדרנו לעץ בבנאי ואימתתי שאכן מתקבלת שגיאה במקרה כזה.