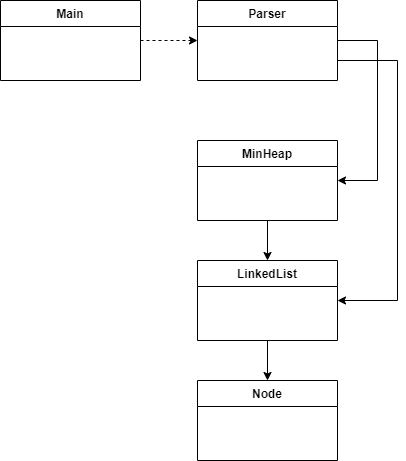
# מבנה התכנית



# המודולים בתכנית

Node – אובייקט פשוט של איבר ברשימה מקושרת

Linked List – רשימה מקושרת בעלת Head ו-Tail, משיקולי יעילות.

Min Heap – ערימת מינימום עבור אובייקטים של רשימה מקושרת.

Parser – מודול Util לצורך קבלת המידע מהמשתמש, ייצור מידע לצורך טסט ולהעברת מידע בין המודולים.

הקוד נכתב ב-Python 3.8 ולא נעשה בו שימוש בחבילות חיצוניות.

אסמכתא לשיקולי יעילות נלקחו מכאן - <https://wiki.python.org/moin/TimeComplexity>

כדי להריץ את הקוד – יש להריץ את הפקודה python –m maman14.main כאשר נמצאים מעל לתיקיה של הקבצים maman14. לחילופין אפשר להריץ את הקוד דרך IDE כרגיל ללא בעיה.

# הסבר לקוד

המספרים הנתונים נלקחים מהמשתמש או מיוצרים רנדומלית. לאחר מכן, הם מועברים לרשימות מקושרות אחד אחד. ראשי הרשימות מועברים לערימת מינימום, כאשר ראש הערימה מצביע למינימום.

הערימה מנוהלת כך שלאחר כל הכנסה של איבר (לסוף הרשימה) מתבצע HeapifyUp שמבצע החלפה בין האיבר האב לשני הילדים שלו, במידת הצורך, כדי לשמור על מאפייני הערימה (האב קטן משני ילדיו) עד להגעה לראש הרשימה או עד שאין צורך בהחלפה.

לאחר כל הוצאת איבר מראש הרשימה, מתבצע HeapifyDown שבמידת האפשר, "מושך" את האיבר הבא מהרשימה בה היה האיבר בראש הרשימה או אם הרשימה ריקה, מבצע חילוף בין האיבר האחרון לראש הרשימה. לאחר שינוי זה, מתבצעות החלפות מראש הרשימה מטה בין האב ושני הבנים שלו, במידת הצורך, כדי לשמור על מאפייני הערימה (האב קטן משני ילדיו), עד להגעה לעלה.

לאחר הכנסת כל הראשים של הרשימות אל הערימה, מתבצעת הוצאה של ראש הערימה והדפסתו, כך שבכל פעם מודפס המינימום מכל איברי הרשימה. מכיוון שנתון שהרשימות ממוינות ומכיוון שקיים לפחות האיבר המינימלי מכל רשימה, כל הוצאה של ראש הערימה תדפיס את המינימום הבא מכל הרשימות.

יעילות הקוד – קליטת הנתונים והכנסת לרשימות תהיה מן הסתם O(n) מכיוון שיש צורך לעבור איבר איבר ולהכניס מהפלט לרשימה.

הוספת איבר לרשימה היא O(1) כיון שלרשימה יש Tail ולכן בכל פעם מתווסף איבר לסוף הרשימה.

הוספת איבר לערימה היא O(1) כאשר כל פעולת Heapify עולה O(log n) כיון שהיא "מטפסת" במעלה הערימה.

הוצאת איבר מהערימה הוא גם O(1) וגם היא דורשת Heapify שעולה O(log n) כיון שהיא "יורדת" על פני גובה הערימה.

# דוגמאות לריצה של הקוד

