```
"""
* Eshan Nahar DSLA1
In second year computer engineering class, group A student's play cricket, group B
students play badminton and group C students play football.
Write a Python program using functions to compute following:
a) List of students who play both cricket and badminton
b) List of students who play either cricket or badminton but not both
c) Number of students who play neither cricket nor badminton
d) Number of students who play cricket and football but not badminton.
"""



def cricket_badminton(cricket, badminton):
    # Returns a list of students who play both cricket and badminton.
    result = []
    for student in cricket:
         if student in badminton:
              result.append(student)
    return result


def either_not_both(cricket, badminton):
    # Returns a list of students who play either cricket or badminton but not both.
    result = []
    for student in cricket:
         if student not in badminton:
```

```
                result.append(student)
    for student in badminton:
          if student not in cricket:
                result.append(student)
    return result


def neither(cricket, badminton, football):
    # Returns the number of students who play neither cricket nor badminton.
    all_students = set()
    for student in cricket:
          all_students.add(student)
    for student in badminton:
          all_students.add(student)
    for student in football:
          all_students.add(student)


    cricket_badminton_students = set(cricket_badminton(cricket, badminton))
    neither_cricket_nor_badminton = all_students - cricket_badminton_students - set(cricket) - set(badminton)
    return len(neither_cricket_nor_badminton)



def cricket_football_not_badminton(cricket, badminton, football):
    # Returns the number of students who play cricket and football but not badminton.
    result = 0
```

```
    for student in cricket:
        if student in football and student not in badminton:
            result += 1
    return result



# Sample data
cricket = ['A1', 'A2', 'A3', 'A4', 'A5']
badminton = ['B1', 'B2', 'B3', 'A3', 'C2']
football = ['B1', 'B2', 'A2', 'C1', 'C2']
print("Students who play both cricket and badminton:")
print(cricket_badminton(cricket, badminton))
print("\nStudents who play either cricket or badminton but not both:")
print(either_not_both(cricket, badminton))
print(f"\nNumber of students who play neither cricket nor badminton: {neither(cricket, badminton, football)}")
print(f"\nNumber of students who play cricket and football but not badminton: {cricket_football_not_badminton(cricket, badminton, football)}")
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A> python A1.py
Students who play both cricket and badminton:
['A3']

Students who play either cricket or badminton but not both:
['A1', 'A2', 'A4', 'A5', 'B1', 'B2', 'B3', 'C2']

Number of students who play neither cricket nor badminton: 1

Number of students who play cricket and football but not badminton: 1
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A>
```

```
"""
* Eshan Nahar DSLA4
4. WAPP that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following: D 100 W 200 (Withdrawal is not allowed if balance is going negative. Write functions for withdraw and deposit) D means deposit while W means withdrawal. Suppose the following input is supplied to the program: D 300, D 300, W 200, D 100 Then output should be 500
"""


def deposit(balance, amount):
  return balance + amount


def withdraw(balance, amount):
  if balance >= amount:
    return balance - amount
  else:
    return balance


transactions = input("Enter the transaction log (EX: D 300, D 300, W 200, D 100): ").split(", ")
balance = 0
for transaction in transactions:
  operation, amount = transaction.split()
  amount = float(amount)
  if operation == "D":
    balance = deposit(balance, amount)
  elif operation == "W":
```

```
    balance = withdraw(balance, amount)


# Print the final balance
print(f"The net amount in the bank account is: {balance}")
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A> python A4.py
Enter the transaction log (EX: D 300, D 300, W 200, D 100): D 200, D 200, W 100
The net amount in the bank account is: 300.0
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A> python A4.py
Enter the transaction log (EX: D 300, D 300, W 200, D 100): W 100
The net amount in the bank account is: 0
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A>
```

```
"""
* Eshan Nahar DSLA2
2. Write a Python program to store marks scored in subject
“Fundamental of Data Structure” by N students in the class. Write
functions to compute following:

    a) The average score of class
    b) Highest score and lowest score of class
    c) Count of students who were absent for the test
    d) Display mark with highest frequency
"""

def average_score(marks):
    # Computes the average score of the class.
    total_score = sum(score for score in marks if score >= 0)
    total_students = sum(1 for score in marks if score >= 0)
    return total_score / total_students

def highest_lowest_score(marks):
    # Returns the highest and lowest scores of the class.
    valid_scores = [score for score in marks if score >= 0]
    return max(valid_scores), min(valid_scores)

def absent_count(marks):
    # Returns the count of students who were absent for the test.
    return sum(1 for score in marks if score < 0)

def most_frequent_mark(marks):
```

```
    # Displays the mark with the highest frequency.
    mark_counts = {}
    for score in marks:
        if score >= 0:
            mark_counts[score] = mark_counts.get(score, 0) + 1

    if not mark_counts:
        return "No valid scores recorded."

    max_count = max(mark_counts.values())
    for mark, count in mark_counts.items():
        if count == max_count:
            return f"The mark with the highest frequency is:
{mark}"

# Example usage
num_students = int(input("Enter the number of students: "))
marks = []
for _ in range(num_students):
    score = int(input(f"Enter the score for student {_+1}: "))
    marks.append(score)

print(f"The average score of the class is:
{average_score(marks):.2f}")
highest, lowest = highest_lowest_score(marks)
print(f"The highest score is: {highest}")
print(f"The lowest score is: {lowest}")
```

```
print(f"The number of students who were absent for the test is:
{absent_count(marks)}")

print(most_frequent_mark(marks))
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A> python A2.py
Enter the number of students: 6
Enter the score for student 1: -1
Enter the score for student 2: 10
Enter the score for student 3: 10
Enter the score for student 4: 11
Enter the score for student 5: 12
Enter the score for student 6: 20
The average score of the class is: 12.60
The highest score is: 20
The lowest score is: 10
The number of students who were absent for the test is: 1
The mark with the highest frequency is: 10
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\A>
```

```
"""
* Eshan Nahar
12. :
  a) WAPP to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using binary search (recursive and non-recursive). Insert friend if not present in phonebook
  b) WAPP to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using Fibonacci search. Insert friend if not present in phonebook.
"""


class PhoneBook:
    def __init__(self):
        self.friends = []


    def insert(self, name, number):
        # Insert the new contact and then sort the list
        self.friends.append((name, number))
        self.insertionSort()


    def insertionSort(self):
        for i in range(1, len(self.friends)):
            key = self.friends[i]
            j = i - 1
            while j >= 0 and key[0] < self.friends[j][0]:
                self.friends[j + 1] = self.friends[j]
                j -= 1
            self.friends[j + 1] = key
```

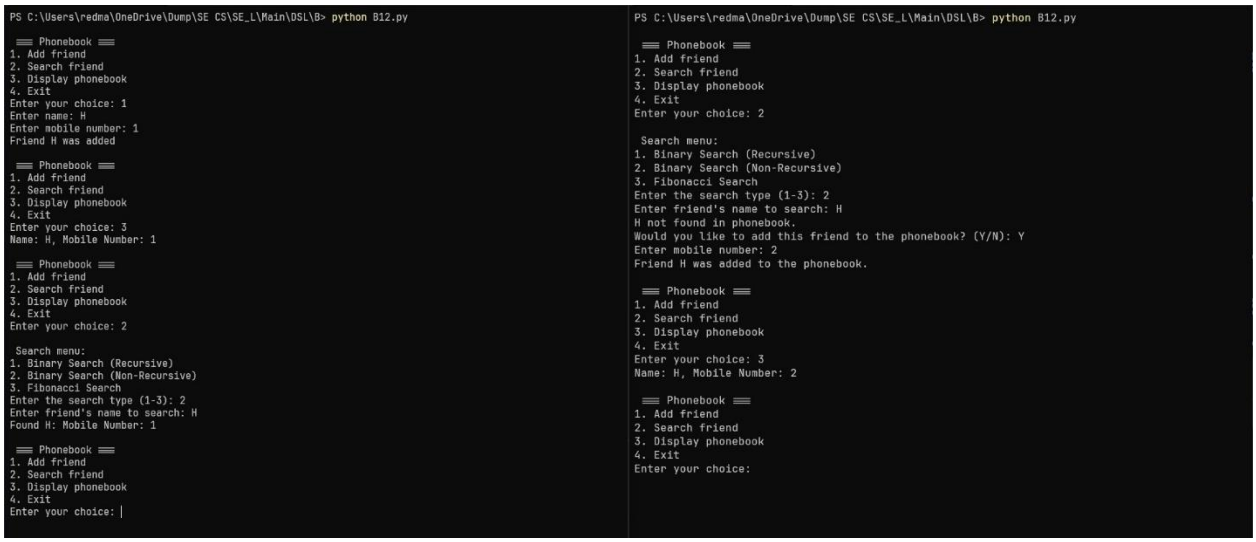
```
def binarySearch(self, name, low, high):
    if low <= high:
        mid = (low + high) // 2
        mid_name = self.friends[mid][0]
        if mid_name == name:
            return mid
        elif name < mid_name:
            return self.binarySearch(name, low, mid - 1)
        else:
            return self.binarySearch(name, mid + 1, high)
    return -1

def binarySearchIt(self, name):
    low, high = 0, len(self.friends) - 1
    while low <= high:
        mid = (low + high) // 2
        mid_name = self.friends[mid][0]
        if mid_name == name:
            return mid
        elif name < mid_name:
            high = mid - 1
        else:
            low = mid + 1
    return -1

def fibonacciSearch(self, name):
```

```
        n = len(self.friends)
        fibm2, fibm1 = 0, 1
        fibm = fibm2 + fibm1
        while fibm < n:
            fibm2, fibm1 = fibm1, fibm
            fibm = fibm1 + fibm2

        offset = -1
        while fibm > 1:
            i = min(offset + fibm2, n - 1)
            if self.friends[i][0] < name:
                fibm, fibm1 = fibm1, fibm - fibm1
                fibm2 = fibm - fibm1
                offset = i
            elif self.friends[i][0] > name:
                fibm, fibm1 = fibm1, fibm1 - fibm2
                fibm2 = fibm - fibm1
            else:
                return i
        if fibm1 and offset < n - 1 and self.friends[n - 1][0] ==
name:
            return n - 1
        return -1

    def display(self):
        if not self.friends:
            print("Phonebook is empty")
```

```
        else:
            for name, number in self.friends:
                print(f"Name: {name}, Mobile Number: {number}")

    def search(self, name, searchType):
        if searchType == 1:
            return self.binarySearch(name, 0, len(self.friends) - 1)
        elif searchType == 2:
            return self.binarySearchIt(name)
        elif searchType == 3:
            return self.fibonacciSearch(name)
        else:
            print("Invalid search selected")
            return None

def main():
    phonebook = PhoneBook()
    while True:
        print("\n === Phonebook ===")
        print("1. Add friend")
        print("2. Search friend")
        print("3. Display phonebook")
        print("4. Exit")
        choice = input("Enter your choice: ").strip()
        if not choice.isdigit():
            print("Invalid input. Please enter a number.")
            continue
```

```
        choice = int(choice)
        if choice == 1:
            name = input("Enter name: ").strip()
            number = input("Enter mobile number: ").strip()
            if not name or not number:
                print("Name and mobile number cannot be empty.")
                continue
            phonebook.insert(name, number)
            print(f"Friend {name} was added")


        elif choice == 2:
            print("\n Search menu:")
            print("1. Binary Search (Recursive)")
            print("2. Binary Search (Non-Recursive)")
            print("3. Fibonacci Search")
            search_type = input("Enter the search type (1-3):
").strip()
            if not search_type.isdigit():
                print("Invalid input. Please enter a number between 1
and 3.")
                continue
            search_type = int(search_type)
            name = input("Enter friend's name to search: ").strip()
            if not name:
                print("Name cannot be empty.")
                continue
            friend_index = phonebook.search(name, search_type)
            if friend_index != -1:
```

```
                print(f"Found {name}: Mobile Number:
{phonebook.friends[friend_index][1]}")
            else:
                print(f"{name} not found in phonebook.")
                add_friend = input("Would you like to add this friend
to the phonebook? (Y/N): ").strip().lower()
                if add_friend == 'Y':
                    number = input("Enter mobile number: ").strip()
                    if not number:
                        print("Mobile number cannot be empty.")
                        continue
                    phonebook.insert(name, number)
                    print(f"Friend {name} was added to the
phonebook.")
                else:
                    print("Friend not added.")
        elif choice == 3:
            phonebook.display()
        elif choice == 4:
            print("Exiting...")
            break
        else:
            print("Invalid choice")

if __name__ == '__main__':
    main()
```



```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\B> python B12.py

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice: 1
Enter name: H
Enter mobile number: 1
Friend H was added

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice: 3
Name: H, Mobile Number: 1

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice: 2

 Search menu:
1. Binary Search (Recursive)
2. Binary Search (Non-Recursive)
3. Fibonacci Search
Enter the search type (1-3): 2
Enter friend's name to search: H
Found H: Mobile Number: 1

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice:
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\B> python B12.py

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice: 2

 Search menu:
1. Binary Search (Recursive)
2. Binary Search (Non-Recursive)
3. Fibonacci Search
Enter the search type (1-3): 2
Enter friend's name to search: H
H not found in phonebook.
Would you like to add this friend to the phonebook? (Y/N): Y
Enter mobile number: 2
Friend H was added to the phonebook.

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice: 3
Name: H, Mobile Number: 2

 == Phonebook ==
1. Add friend
2. Search friend
3. Display phonebook
4. Exit
Enter your choice:
```

```
"""
* Eshan Nahar
14. WAPP to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using
  a) Selection Sort
  b) Bubble sort and display top five scores

"""


def selectionsort(arr):
  n = len(arr)
  for i in range(n-1):
    min_idx = i
    for j in range(i+1, n):
      if arr[j] > arr[min_idx]:
        min_idx = j
    arr[i], arr[min_idx] = arr[min_idx], arr[i]


def bubblesort(arr):
  n = len(arr)
  for i in range(n-1):
    for j in range(n-1-i):
      if arr[j] < arr[j+1]:
        arr[j], arr[j+1] = arr[j+1], arr[j]


def main():
  student_count = int(input("Enter the number of students: "))
```

```
  percentages = []
  for i in range(student_count):
    percentage = float(input(f"Enter the percentage for student {i + 1}: "))
    percentages.append(percentage)
  ch = int(input("=== Choose a sorting method ===\n1. Selection Sort\n2. Bubble Sort\n"))
  if ch == 1:
    selectionsort(percentages)
  elif ch == 2:
    bubblesort(percentages)
  else:
    print("Invalid choice")
  print("Top 5 scores are:")
  for i, percentage in enumerate(percentages[:5], 1):
    print(f"{i}: {percentage:.2f}%")

if __name__ == "__main__":
  main()
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\B> python B14.py
Enter the number of students: 10
Enter the percentage for student 1: 3
Enter the percentage for student 2: 6
Enter the percentage for student 3: 1
Enter the percentage for student 4: 5
Enter the percentage for student 5: 3
Enter the percentage for student 6: 7
Enter the percentage for student 7: 9
Enter the percentage for student 8: 10
Enter the percentage for student 9: 3
Enter the percentage for student 10: 1
≡≡≡ Choose a sorting method ≡≡≡
1. Selection Sort
2. Bubble Sort
2
Top 5 scores are:
1: 10.00%
2: 9.00%
3: 7.00%
4: 6.00%
5: 5.00%
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\B>
```

```
"""
* Eshan Nahar
18. Write Python program to store 10th class percentage of students
in array. Write function for sorting array of floating point numbers
in ascending order using radix sort and display top five scores
"""


def countingsort(num, k):
    n = len(num)
    res = [0] * n
    c = [0] * 10 # Count
    for i in range(0, n):
        temp = num[i] // k # Index
        c[temp % 10] += 1 # Populate count array
    for i in range(1,10):
        c[i] += c[i-1] # Cumulative count array
    i = n-1 # Start from end
    while i>=0: # Assign numbers
        temp = num[i] // k
        res[c[temp % 10] - 1] = num[i]
        c[temp % 10] -= 1
        i = i-1
    for i in range(0, n):
        num[i] = res[i] # Copy to original array
    print(num)
    # Top Five Scores
    for i in range(len(num)):
        x = num[i]
```

```
        x /= 10
        num2[i] = x
    print(num2)

def radixsort(num):
    print(num)
    maximum = max(num)
    print(maximum)
    n = 1
    while maximum // n > 0:
        countingsort(num, n)
        n *= 10 # Increase exponent

def countDigits(n): # Helper function (Unused)
    cnt = 0
    countAfterDecimal = str(n)[::-1].find('.')
    while n>0:
        cnt += 1
        n = n // 10
    return cnt + countAfterDecimal

num = []
num1 = []
num2 = []

M = int(input("Enter the number of students "))
for i in range(M):
```

```
    num2.append(0)
    per = float(input("Enter the percentage marks "))
    num.append(per)

num1 = [0] * len(num)
for i in range(0, len(num)):
    num1[i] = int(num[i]*10) # Convert to integer

radixsort(num1)
num3 = num2[::-1]
print("Top 5 scores are: ")
for i in range(0,5):
    print(num3[i])
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\B> python B18.py
Enter the number of students 5
Enter the percentage marks 25.3
Enter the percentage marks 64.2
Enter the percentage marks 16.3
Enter the percentage marks 78.4
Enter the percentage marks 25.8
[642, 253, 163, 784, 258]
[64.2, 25.3, 16.3, 78.4, 25.8]
[642, 253, 258, 163, 784]
[64.2, 25.3, 25.8, 16.3, 78.4]
[163, 253, 258, 642, 784]
[16.3, 25.3, 25.8, 64.2, 78.4]
Top 5 scores are:
78.4
64.2
25.8
25.3
16.3
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\B>
```

```
// * Eshan Nahar

// 19. Department of Computer Engineering has student's club named
'Pinnacle Club'. Students of second, third and final year of
department can be granted membership on request. Similarly one may
cancel the membership of club. First node is reserved for president
of club and last node is reserved for secretary of club. WCP to
maintain club member‘s information using singly linked list. Store
student PRN and Name. Write functions to:


//   a) Add and delete the members as well as president or even
secretary.

//   b) Compute total number of members of club

//   c) Display members

//   d) Two linked lists exists for two divisions. Concatenate two
lists.


#include <iostream>

#include <limits>

#include <string>

using namespace std;


enum Role { PRESIDENT, SECRETARY, MEMBER };

// Node structure definition

struct node {

  string name;

  int PRN;

  Role role;

  node *next;

  node(string nm, int prn, Role r)

      : name(nm), PRN(prn), role(r), next(nullptr) {}
```

```
};

// Linked list class definition
class ll {
  node *head;
  node *tail;

 public:
  ll() : head(nullptr), tail(nullptr) {}
  ~ll() {
    while(head) { removePresident(); }
  }

  void create();
  void display();
  void addPresident();
  void addSecretary();
  void addMember();
  void removePresident();
  void deleteSecretary();
  void deleteMember(int prn);
  int computeTotal();
  void concatLists(ll &other);

 private:
  bool isValidPRN(int prn);
  node *createNode(Role role);
```

```
};

bool ll::isValidPRN(int prn) { return prn > 0; }

node *ll::createNode(Role role) {
  string name;
  int prn;

  cout << "Enter PRN number for "
       << ((role == PRESIDENT)   ? "President"
           : (role == SECRETARY) ? "Secretary"
                                 : "Member")
       << ": ";
  while(!(cin >> prn) || !isValidPRN(prn)) {
    cout << "Invalid PRN. Please enter a positive integer: ";
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
  }

  cout << "Enter name: ";
  cin >> name;

  return new node(name, prn, role);
}

void ll::create() {
  head = createNode(PRESIDENT);
```

```
  tail = head;
  cout << "List created with President: " << head->name << endl;
}

void ll::display() {
  node *temp = head;
  if(!temp) {
    cout << "List is empty." << endl;
    return;
  }
  cout << "Club Members:\n";
  while(temp) {
    switch(temp->role) {
      case PRESIDENT:
        cout << "President: " << temp->name << ", PRN: " << temp->PRN
<< endl;
        break;
      case SECRETARY:
        cout << "Secretary: " << temp->name << ", PRN: " << temp->PRN
<< endl;
        break;
      case MEMBER:
        cout << "Member: " << temp->name << ", PRN: " << temp->PRN <<
endl;
        break;
    }
    temp = temp->next;
  }
```

```
}

void ll::addPresident() {
  node *newPresident = createNode(PRESIDENT);
  newPresident->next = head;
  head = newPresident;
  if(!tail) tail = newPresident;
}

void ll::addSecretary() {
  node *newSecretary = createNode(SECRETARY);
  if(!head) {
    head = newSecretary;
    tail = newSecretary;
  } else {
    tail->next = newSecretary;
    tail = newSecretary;
  }
}

void ll::addMember() {
  node *newMember = createNode(MEMBER);
  if(!head) {
    head = newMember;
    tail = newMember;
  } else {
    tail->next = newMember;
```

```
    tail = newMember;
  }
}

void ll::removePresident() {
  if(!head) {
    cout << "No president to remove." << endl;
    return;
  }
  node *temp = head;
  head = head->next;
  delete temp;
  cout << "President removed." << endl;
}

void ll::deleteSecretary() {
  if(!head || head->role != SECRETARY) {
    cout << "No secretary to remove." << endl;
    return;
  }
  if(head == tail) { // Only one member
    delete head;
    head = tail = nullptr;
    cout << "Secretary removed." << endl;
    return;
  }
```

```
  node *temp = head;
  while(temp->next && temp->next->role != SECRETARY) { temp = temp->next; }
  if(!temp->next) {
    cout << "No secretary to remove." << endl;
    return;
  }
  node *secretaryToRemove = temp->next;
  temp->next = secretaryToRemove->next;
  if(secretaryToRemove == tail) { tail = temp; }
  delete secretaryToRemove;
  cout << "Secretary removed." << endl;
}

void ll::deleteMember(int prn) {
  node *temp = head;
  node *prev = nullptr;
  while(temp && temp->PRN != prn) {
    prev = temp;
    temp = temp->next;
  }
  if(!temp) {
    cout << "Member with PRN " << prn << " not found." << endl;
    return;
  }

  if(prev) prev->next = temp->next;
```

```
  else head = temp->next;

  if(temp == tail) { tail = prev; }
  delete temp;
  cout << "Member with PRN " << prn << " removed." << endl;
}

int ll::computeTotal() {
  node *temp = head;
  int count = 0;
  while(temp) {
    count++;
    temp = temp->next;
  }
  return count;
}

void ll::concatLists(ll &other) {
  if(!head) head = other.head;
  else tail->next = other.head;
  if(other.tail) { tail = other.tail; }
  other.head = nullptr;
  other.tail = nullptr;
  cout << "Lists concatenated." << endl;
}

int main() {
```

```
  ll list1, list2;
  int choice, listChoice;

  do {
    cout << "\n=== Select Option ===\n1. List 1\n2. List 2\n3. Concatenate "
            "Lists\n0. Exit\nEnter choice: ";
    cin >> listChoice;
    ll *currentList = (listChoice == 1)   ? &list1
                    : (listChoice == 2) ? &list2
                                        : nullptr;

    if(listChoice == 3) {
      cout << "Concatenating List 1 and List 2..." << endl;
      list1.concatLists(list2);
    } else if(!currentList) {
      cout << "Invalid list choice. Please try again." << endl;
      continue;
    }

    if(listChoice == 0) break;

    cout << "\n1. Create\n2. Add President\n3. Add Secretary\n4. Add "
            "Member\n5. Display\n"
         << "6. Remove President\n7. Remove Secretary\n8. Remove Member\n9. "
            "Total Members\n"
```

```
     << "0. Exit";
do {
  cout << "\nEnter your choice: ";
  cin >> choice;
  switch(choice) {
    case 1: currentList->create(); break;
    case 2: currentList->addPresident(); break;
    case 3: currentList->addSecretary(); break;
    case 4: currentList->addMember(); break;
    case 5: currentList->display(); break;
    case 6: currentList->removePresident(); break;
    case 7: currentList->deleteSecretary(); break;
    case 8: {
      int prn;
      cout << "Enter PRN of member to be deleted: ";
      cin >> prn;
      currentList->deleteMember(prn);
      break;
    }
    case 9:
      cout << "Total members (including President & Secretary): "
           << currentList->computeTotal() << endl;
      break;
    case 0: break;
    default: cout << "Invalid choice." << endl;
  }
} while(choice != 0);
```

```
} while(listChoice != 0);


return 0;
```

```
}
```

```
C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\C>g++ C19Format.cpp && .\a.exe

=== Select Option ===
1. List 1
2. List 2
3. Concatenate Lists
0. Exit
Enter choice: 1

1. Create
2. Add President
3. Add Secretary
4. Add Member
5. Display
6. Remove President
7. Remove Secretary
8. Remove Member
9. Total Members
0. Exit
Enter your choice: 1
Enter PRN number for President: 1
Enter name: President
List created with President: President

Enter your choice: 3
Enter PRN number for Secretary: 2
Enter name: Secretary

Enter your choice: 4
Enter PRN number for Member: 666
Enter name: Eshan

Enter your choice: 4
Enter PRN number for Member: 777
Enter name: Something

Enter your choice: 5
Club Members:
President: President, PRN: 1
Secretary: Secretary, PRN: 2
Member: Eshan, PRN: 666
Member: Something, PRN: 777

Enter your choice: 9
Total members (including President & Secretary): 4

Enter your choice: 6
```

```
// * Eshan Nahar
// 22. Second year Computer Engineering class, set A of students like
Vanilla Ice-cream and set B of students like butterscotch ice-cream.
WCP to store two sets using linked list. compute and display-

//   a) Set of students who like both vanilla and butterscotch
//   b) Set of students who like either vanilla or butterscotch or not
both
//   c) Number of students who like neither vanilla nor

#include <iostream>
#include <string>
using namespace std;

struct Node {
  string studentName;
  Node* next;
};

Node* createNode(const string& studentName) {
  Node* newNode = new Node();
  newNode->studentName = studentName;
  newNode->next = nullptr;
  return newNode;
}

void insertEnd(Node*& head, const string& studentName) {
  Node* newNode = createNode(studentName);
```

```
  if (head == nullptr) head = newNode;
  else {
    Node* temp = head;
    while (temp->next != nullptr) temp = temp->next;
    temp->next = newNode;
  }
}

void displayList(Node* head) {
  Node* temp = head;
  if (temp == nullptr) cout << "No students in the list." << endl;
  else {
    while (temp != nullptr) {
      cout << temp->studentName << " ";
      temp = temp->next;
    }
    cout << endl;
  }
}

void toLower(string& str) {
  for (char& c : str)
    if (c >= 'A' && c <= 'Z')
      c = c + ('a' - 'A');
}

// Function to check if a studentName exists in the list
```

```
bool exists(Node* head, const string& studentName) {
  Node* temp = head;
  string lowerName = studentName;
  toLower(lowerName);
  while (temp != nullptr) {
    string lowerTempName = temp->studentName;
    toLower(lowerTempName);
    if (lowerTempName == lowerName) return true;
    temp = temp->next;
  }
  return false;
}

Node* intersection(Node* headA, Node* headB) {
  Node* result = nullptr;
  Node* tempA = headA;
  while (tempA != nullptr) {
    if (exists(headB, tempA->studentName))
      insertEnd(result, tempA->studentName);
    tempA = tempA->next;
  }
  return result;
}

// Function to compute the union of two sets
Node* unionSets(Node* headA, Node* headB) {
  Node* result = nullptr;
```

```
  Node* tempA = headA;
  Node* tempB = headB;
  while (tempA != nullptr) {
    insertEnd(result, tempA->studentName);
    tempA = tempA->next;
  }

  // Add all elements from set B that are not in set A
  while (tempB != nullptr) {
    if (!exists(result, tempB->studentName))
      insertEnd(result, tempB->studentName);
    tempB = tempB->next;
  }

  return result;
}

Node* difference(Node* headA, Node* headB) {
  Node* result = nullptr;
  Node* tempA = headA;
  while (tempA != nullptr) {
    if (!exists(headB, tempA->studentName))
      insertEnd(result, tempA->studentName);
    tempA = tempA->next;
  }
  return result;
}
```

```
// Function to compute the symmetric difference (A Δ B)
Node* symmetricDifference(Node* headA, Node* headB) {
  Node* unionAB = unionSets(headA, headB);
  Node* intersectionAB = intersection(headA, headB);
  return difference(unionAB, intersectionAB);
}

// Function to count the number of nodes in a list
int countNodes(Node* head) {
  int count = 0;
  Node* temp = head;
  while (temp != nullptr) {
      count++;
      temp = temp->next;
  }
  return count;
}

// Function to get the number of students who like neither vanilla nor
butterscotch
int countNeither(Node* vanillaList, Node* butterscotchList, int
totalStudents) {
  Node* unionAB = unionSets(vanillaList, butterscotchList);
  int countAB = countNodes(unionAB);
  return totalStudents - countAB;
}
```

```
int main() {
  Node* vanillaList = nullptr;
  Node* butterscotchList = nullptr;

  int choice, totalStudents;
  string studentName;
  cout << "Enter the total number of students: ";
  cin >> totalStudents;
  cin.ignore();
  cout << "Enter the students who like vanilla (End with empty
line):\n";
  while (getline(cin, studentName) && !studentName.empty())
      insertEnd(vanillaList, studentName);
  cout << "Enter the students who like butterscotch (End with empty
line):\n";
  while (getline(cin, studentName) && !studentName.empty())
      insertEnd(butterscotchList, studentName);

  do {
    cout << "\nMenu:\n";
    cout << "1. Display set of students who like both vanilla and
butterscotch\n";
    cout << "2. Display set of students who like either vanilla or
butterscotch but not both\n";
    cout << "3. Display number of students who like neither vanilla
nor butterscotch\n";
    cout << "4. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;
```

```
    cin.ignore(); // To ignore the newline character left by cin


    switch (choice) {
      case 1:
        Node* both = intersection(vanillaList, butterscotchList);
        cout << "Students who like both vanilla and butterscotch: ";
        displayList(both);
        break;
      case 2:
        Node* eitherNotBoth = symmetricDifference(vanillaList,
butterscotchList);
        cout << "Students who like either vanilla or butterscotch but
not both: ";
        displayList(eitherNotBoth);
          break;
      case 3:
        int neitherCount = countNeither(vanillaList, butterscotchList,
totalStudents);
        cout << "Number of students who like neither vanilla nor
butterscotch: " << neitherCount << endl;
        break;
      case 4:
        cout << "Exiting...\n";
        break;
      default: cout << "Invalid choice, please try again.\n";
    }
  } while (choice != 4);
  return 0;
```

}

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\C> g++ C22.cpp -o o
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\C> .\o.exe
Enter the total number of students: 5
Enter the students who like vanilla (End with empty line):
Alice
Bob
Charlie

Enter the students who like butterscotch (End with empty line):
David
Eve
Charlie


Menu:
1. Display set of students who like both vanilla and butterscotch
2. Display set of students who like either vanilla or butterscotch but not both
3. Display number of students who like neither vanilla nor butterscotch
4. Exit
Enter your choice: 1
Students who like both vanilla and butterscotch: Charlie

Menu:
1. Display set of students who like both vanilla and butterscotch
2. Display set of students who like either vanilla or butterscotch but not both
3. Display number of students who like neither vanilla nor butterscotch
4. Exit
Enter your choice: 2
Students who like either vanilla or butterscotch but not both: Alice Bob David Eve

Menu:
1. Display set of students who like both vanilla and butterscotch
2. Display set of students who like either vanilla or butterscotch but not both
3. Display number of students who like neither vanilla nor butterscotch
4. Exit
Enter your choice: 3
Number of students who like neither vanilla nor butterscotch: 0

Menu:
1. Display set of students who like both vanilla and butterscotch
2. Display set of students who like either vanilla or butterscotch but not both
3. Display number of students who like neither vanilla nor butterscotch
4. Exit
Enter your choice: 4
Exiting...
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\C>
```

```
/**
 * * Eshan Nahar
 * * Literally leetcode 20
 * In any language program mostly syntax error occurs due to unbalancing
delimiter such as (),{},[]. Write C++ program using stack to check whether given
expression is well parenthesized or not
 */

#include <iostream>
#include <string>
using namespace std;

class Stack {
private:
  char *arr;
  int top;
  int capacity;

public:
  Stack(int size) {
    capacity = size;
    arr = new char[capacity];
    top = -1;
  }

  ~Stack() { delete[] arr; }
```

```
  bool isEmpty() const { return top == -1; }
  void push(char c) {
    if (top >= capacity - 1) throw overflow_error("Stack overflow");
    arr[++top] = c;
  }

  char pop() {
    if (top < 0)  throw underflow_error("Stack underflow");
    return arr[top--];
  }

  char peek() const {
    if (top < 0) throw underflow_error("Stack is empty");
    return arr[top];
  }

};

bool isWellParenthesized(const string &expression) {
  Stack s(expression.size());
  for (char ch : expression) {
    switch (ch) {
      case '(':
      case '{':
      case '[': s.push(ch); break;
      case ')':
      case '}':
```

```
      case ']':
        if (s.isEmpty() || s.peek() != ch - 1) return false;
        s.pop();
        break;
    }
  }

  return s.isEmpty();
}

int main() {
  string expression;
  cout << "Enter the expression: ";
  getline(cin, expression);
  if (isWellParenthesized(expression)) cout << "The expression is
well-parenthesized.\n";
  else cout << "The expression is not well-parenthesized.\n";
  return 0;
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\D> g++ D26Pure.cpp -o o
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\D> .\o.exe
Enter the expression: {(})
The expression is not well-parenthesized.
```

```
/**
 * * Eshan Nahar
 * Implement C++ program for expression conversion as infix to postfix
and its
evaluation using stack based on given conditions:

  1. Operands and operator, both must be single character.
  2. Input Postfix expression must be in a desired format.
  3. Only '+', '-', '*' and '/ ' operators are expected.
 */

#include <bits/stdc++.h>
using namespace std;

class Stack {
private:
  string stack;

public:
  void push(char c) { stack += c; }
  char pop() {
    if (stack.empty()) { throw underflow_error("Stack underflow"); }
    char top = stack.back();
    stack.pop_back();
    return top;
  }
```

```
  char peek() const {
    if (stack.empty()) { throw underflow_error("Stack is empty"); }
    return stack.back();
  }

  bool isEmpty() const { return stack.empty(); }
  bool isDigit(char ch) const { return ch >= '0' && ch <= '9'; }
};

int precedence(char op) {
  switch (op) {
    case '+':
    case '-': return 1;
    case '*':
    case '/': return 2;
    default: return 0;
  }
}

string infixToPostfix(const string &infix) {
  Stack s;
  string postfix;

  for (char ch : infix) {
    if (ch >= '0' && ch <= '9') postfix += ch;
    else if (ch == '(') s.push(ch);
    else if (ch == ')') {
```

```
      while (!s.isEmpty() && s.peek() != '(') {
        postfix += s.pop();
      }
      if (!s.isEmpty()) s.pop();
    } else {
      while (!s.isEmpty() && precedence(s.peek()) >= precedence(ch)) {
        postfix += s.pop();
      }
      s.push(ch);
    }
  }

  while (!s.isEmpty()) postfix += s.pop();
  return postfix;
}

int evaluatePostfix(const string &postfix) {
  Stack s;
  for (char ch : postfix) {
    if (ch >= '0' && ch <= '9') s.push(ch);
    else {
      int right = s.pop() - '0';
      int left = s.pop() - '0';

      switch (ch) {
        case '+': s.push((left + right) + '0'); break;
        case '-': s.push((left - right) + '0'); break;
```

```
        case '*': s.push((left * right) + '0'); break;
        case '/': s.push((left / right) + '0'); break;
      }
    }
  }

  return s.pop() - '0';
}

int main() {
  string infix, postfix;
  cout << "Enter the infix expression: ";
  getline(cin, infix);
  postfix = infixToPostfix(infix);
  cout << "Postfix expression: " << postfix << endl;
  int result = evaluatePostfix(postfix);
  cout << "Evaluation of postfix expression: " << result << endl;

  return 0;
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\D> g++ D27Pure.cpp -o o
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\D> .\o.exe
Enter the infix expression: (3+5)*2
Postfix expression: 35+2*
Evaluation of postfix expression: 16
```

```
/**
 * * Eshan Nahar
 * Queues are frequently used in computer programming, and a typical example is
 the creation of a job queue by an operating system. If the operating system
does not use priorities, then the jobs are processed in the order they enter
the system. Write C++ program for simulating job queue. Write functions to add
job and delete job from queue.

*/

#include <iostream>
#define MAX 10
using namespace std;

struct QueueData {
  int data[MAX];
  int front, rear;
};

class Queue {
private:
  QueueData q;

public:
```

```
  Queue() { q.front = q.rear = -1; }
  bool isEmpty() const { return q.front == q.rear; }
  bool isFull() const { return q.rear == MAX - 1; }
  void enqueue(int x);
  int dequeue();
  void display() const;
};

void Queue::enqueue(int x) {
  if (isFull()) {
    cout << "Queue overflow\n\n";
    return;
  }
  q.data[++q.rear] = x;
}

int Queue::dequeue() {
  if (isEmpty()) {
    cout << "Queue underflow\n\n";
    return -1; // Indicate error condition
  }
  return q.data[++q.front];
}

void Queue::display() const {
  if (isEmpty()) {
    cout << "Queue is empty!!!\n\n";
```

```
    return;
  }

  cout << "\nQueue contains: \n";
  for (int i = q.front + 1; i <= q.rear; ++i) {
    cout << q.data[i] << " ";
  }
  cout << endl;
}

int main() {
  Queue obj;
  int choice, x;
  do {
    cout << "\n1. Insert Job\n2. Delete Job\n3. Display\n4.
Exit\nEnter your choice: ";
    cin >> choice;
    switch (choice) {
      case 1:
        cout << "Enter data: ";
        cin >> x;
        obj.enqueue(x);
        break;
      case 2:
        x = obj.dequeue();
        if (x != -1) { cout << "Deleted Element = " << x << endl; }
        obj.display();
```

```
        break;
      case 3: obj.display(); break;
      case 4: cout << "Exiting Program....." << endl; break;
      default: cout << "Invalid choice, please try again." << endl;
break;
    }
  } while (choice != 4);


  return 0;
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E> g++ E29.cpp -o o
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E> .\o.exe

1. Insert Job
2. Delete Job
3. Display
4. Exit
Enter your choice: 1
Enter data: 1

1. Insert Job
2. Delete Job
3. Display
4. Exit
Enter your choice: 3

Queue contains:
1

1. Insert Job
2. Delete Job
3. Display
4. Exit
Enter your choice: 1
Enter data: 2

1. Insert Job
2. Delete Job
3. Display
4. Exit
Enter your choice: 2
Deleted Element = 1

Queue contains:
2

1. Insert Job
2. Delete Job
3. Display
4. Exit
Enter your choice: 4
Exiting Program.....
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E>
```

```
 /**
 * * Eshan Nahar
 * A double-ended queue (deque) is a linear list in which additions and
 deletions may be made at either end. Obtain a data representation mapping a
deque into a one- dimensional array. Write C++ program to simulate deque with
functions to add and delete elements from either end of the deque.

*/

#include <iostream>
#include <string>
using namespace std;

const int MAX_SIZE = 100;

class Deque {
private:
  int arr[MAX_SIZE];
  int front;
  int rear;
  int size;

public:
  Deque() : front(MAX_SIZE / 2), rear(MAX_SIZE / 2), size(0) {}
```

```
bool isEmpty() const { return size == 0; }
bool isFull() const { return size == MAX_SIZE; }

void addFront(int value) {
  if (isFull()) {
    cout << "Error: Deque is full." << endl;
    return;
  }
  front = (front - 1 + MAX_SIZE) % MAX_SIZE;
  arr[front] = value;
  ++size;
  cout << "Added " << value << " to the front." << endl;
}

void addRear(int value) {
  if (isFull()) {
    cout << "Error: Deque is full." << endl;
    return;
  }
  arr[rear] = value;
  rear = (rear + 1) % MAX_SIZE;
  ++size;
  cout << "Added " << value << " to the rear." << endl;
}

void removeFront() {
  if (isEmpty()) {
```

```
    cout << "Error: Deque is empty." << endl;
    return;
  }
  cout << "Removed " << arr[front] << " from the front." << endl;
  front = (front + 1) % MAX_SIZE;
  --size;
}

void removeRear() {
  if (isEmpty()) {
    cout << "Error: Deque is empty." << endl;
    return;
  }
  rear = (rear - 1 + MAX_SIZE) % MAX_SIZE;
  cout << "Removed " << arr[rear] << " from the rear." << endl;
  --size;
}

void display() const {
  if (isEmpty()) {
    cout << "Deque is empty." << endl;
    return;
  }
  cout << "Deque contents:\n";
  int current = front;
  for (int i = 0; i < size; ++i) {
    cout << arr[current] << " ";
```

```
      current = (current + 1) % MAX_SIZE;
    }
    cout << endl;
  }
};

// Function to display the menu
void printMenu() {
  cout << "Deque Menu:\n";
  cout << "1. Add to front\n";
  cout << "2. Add to rear\n";
  cout << "3. Remove from front\n";
  cout << "4. Remove from rear\n";
  cout << "5. Display all elements\n";
  cout << "6. Exit\n";
  cout << "Enter your choice: ";
}

int main() {
  Deque deque;
  int choice;
  int value;
  while (true) {
    printMenu();
    cin >> choice;
    switch (choice) {
      case 1:
```

```
        cout << "Enter value to add to the front: ";
        cin >> value;
        deque.addFront(value);
        break;

      case 2:
        cout << "Enter value to add to the rear: ";
        cin >> value;
        deque.addRear(value);
        break;

      case 3: deque.removeFront(); break;
      case 4: deque.removeRear(); break;
      case 5: deque.display(); break;
      case 6: cout << "Exiting...\n"; return 0;
      default: cout << "Invalid choice, please try again.\n"; break;
    }
  }
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E> g++ E31.cpp -o o
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E> .\o.exe
Deque Menu:
1. Add to front
2. Add to rear
3. Remove from front
4. Remove from rear
5. Display all elements
6. Exit
Enter your choice: 1
Enter value to add to the front: 5
Added 5 to the front.
Deque Menu:
1. Add to front
2. Add to rear
3. Remove from front
4. Remove from rear
5. Display all elements
6. Exit
Enter your choice: 2
Enter value to add to the rear: 1
Added 1 to the rear.
Deque Menu:
1. Add to front
2. Add to rear
3. Remove from front
4. Remove from rear
5. Display all elements
6. Exit
Enter your choice: 5
Deque contents:
5 1
Deque Menu:
1. Add to front
2. Add to rear
3. Remove from front
4. Remove from rear
5. Display all elements
6. Exit
Enter your choice: 3
Removed 5 from the front.
Deque Menu:
1. Add to front
2. Add to rear
3. Remove from front
4. Remove from rear
5. Display all elements
6. Exit
Enter your choice: 5
Deque contents:
1
Deque Menu:
1. Add to front
2. Add to rear
```

```
/**
 * * Eshan Nahar
 * Pizza parlor accepting maximum M orders. Orders are served in first come
 first served basis. Order once placed cannot be cancelled. Write C++ program to
simulate the system using circular queue using array.

*/

#include <iostream>
#include <string>
using namespace std;

const int MAX_SIZE = 100;
class CircularQueue {
private:
  string orders[MAX_SIZE];
  int front;
  int rear;
  int size;

public:
  CircularQueue() : front(0), rear(0), size(0) {}
  bool isEmpty() const { return size == 0; }
  bool isFull() const { return size == MAX_SIZE; }

  void addOrder(const string &order) {
```

```
  if (isFull()) {
    cout << "Error: Queue is full. Cannot add new orders." << endl;
    return;
  }
  orders[rear] = order;
  rear = (rear + 1) % MAX_SIZE;
  ++size;
  cout << "Order added: " << order << endl;
}

void serveOrder() {
  if (isEmpty()) {
    cout << "Error: Queue is empty. No orders to serve." << endl;
    return;
  }
  cout << "Order served: " << orders[front] << endl;
  front = (front + 1) % MAX_SIZE;
  --size;
}

void displayOrders() const {
  if (isEmpty()) {
    cout << "No orders in the queue." << endl;
    return;
  }
  cout << "Current orders in the queue:\n";
  int i = front;
```

```
    for (int count = 0; count < size; ++count) {
      cout << orders[i] << " ";
      i = (i + 1) % MAX_SIZE;
    }
    cout << endl;
  }
};

// Function to display the menu
void printMenu() {
  cout << "Pizza Parlor Menu:\n";
  cout << "1. Add an order\n";
  cout << "2. Serve an order\n";
  cout << "3. Display all orders\n";
  cout << "4. Exit\n";
  cout << "Enter your choice: ";
}

int main() {
  CircularQueue queue;
  int choice;
  string order;

  while (true) {
    printMenu();
    cin >> choice;
    cin.ignore();
```

```
    switch (choice) {
      case 1:
        cout << "Enter order description: ";
        getline(cin, order);
        queue.addOrder(order);
        break;

      case 2: queue.serveOrder(); break;
      case 3: queue.displayOrders(); break;
      case 4: cout << "Exiting...\n"; return 0;
      default: cout << "Invalid choice, please try again.\n"; break;
    }
  }
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E> g++ E32.cpp -o o
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E> .\o.exe
Pizza Parlor Menu:
1. Add an order
2. Serve an order
3. Display all orders
4. Exit
Enter your choice: 1
Enter order description: Order
Order added: Order
Pizza Parlor Menu:
1. Add an order
2. Serve an order
3. Display all orders
4. Exit
Enter your choice: 2
Order served: Order
Pizza Parlor Menu:
1. Add an order
2. Serve an order
3. Display all orders
4. Exit
Enter your choice: 3
No orders in the queue.
Pizza Parlor Menu:
1. Add an order
2. Serve an order
3. Display all orders
4. Exit
Enter your choice: 4
Exiting...
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\DSL\E>
```