2. АЛГОРИТМЫ ОБРАБОТКИ СПИСКОВ И ХЕШИРОВАНИЯ

2.1. Задания по обработке списков

1. Разработать функции для создания стека:
- заполнение стека возрастающими числами;
- заполнение стека убывающими числами;
- заполнение стека случайными числами;

2. Разработать функции для создания очереди:
- заполнение очереди возрастающими числами;
- заполнение очереди убывающими числами;
- заполнение очереди случайными числами;

3. Разработать функции для работы со списком:
- печать элементов списка;
- подсчет контрольной суммы элементов списка;
- подсчет количества серий в списке.

4*. Разработать функцию для удаление всех элементов из списка.

5*. Разработать рекурсивную функцию печати элементов списка в прямом и обратном порядке.

2.2. Задания по сортировке прямого слияния MergeSort

1. Для набора из 12 первых последовательных символов ФИО студента выполнить вручную сортировку прямого слияния MergeSort. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограмму расщепления списка (n>=20) на два списка (через один элемент), для проверки правильности расщепления вывести на экран списки и количество элементов в них.

3. Разработать подпрограмму слияния серий. Для проверки правильности слияния вывести на экран количество серий и контрольную сумму для всех списков. Предусмотреть подсчет фактического количества сравнений и перемещений (из стека в очередь) Сф и Мф, сравнить с теоретическими оценками С и М.

4. Разработать подпрограмму сортировки списка методом прямого слияния (MergeSort). Предусмотреть подсчет фактического количества сравнений и перемещений (из стека в очередь) Сф и Мф, сравнить с теоретическими оценками С и М.

5. Сравнить время работы сортировки прямого слияния MergeSort на массивах убывающих, возрастающих и случайных чисел (по сумме Мф+Сф) и сделать вывод о зависимости (или независимости) метода MergeSort от исходной упорядоченности массива.

Составить таблицу:

Трудоемкость сортировки прямого слияния

N	M+C теоретич.	$M_{факт}+C_{факт}$		
		Убыв.	Случ.	Возр.
100				
200				
300				
400				
500				

6*. Построить на экране в одной координатной плоскости графики зависимости трудоемкости (Мф+Сф) от размера массива n для пирамидальной сортировки и метода Хоара (для массива случайных чисел), и метода прямого слияния (для списка случайных чисел).

2.3. Задания по цифровой сортировке DigitalSort

1. Для набора из 12 трехзначных чисел в 6-ичной системе счисления выполнить вручную цифровую сортировку DigitalSort. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограмму цифровой сортировки DigitalSort для списка целых чисел по возрастанию. Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в списке. Предусмотреть подсчет фактического количества перемещений (из стека в очередь) Мф, сравнить с теоретическими оценками М.

3. Применить DigitalSort для сортировки списка двухбайтовых и четырехбайтовых целых чисел в прямом и обратном порядке.

4. Сравнить время работы цифровой сортировки DigitalSort на списках убывающих, возрастающих и случайных чисел (по сумме Мф) и сделать вывод о зависимости (или независимости) сортировки DigitalSort от исходной упорядоченности списка.

Составить таблицу:

Трудоемкость цифровой сортировки DigitalSort

N	M	$M_{факт}$

	теоретич.	Убыв.	Случ.	Возр.
100				
200				
300				
400				
500				

5*. Построить на экране в одной координатной плоскости графики зависимости трудоемкости от размера массива или списка для пирамидальной сортировки, метода Хоара, метода прямого слияния и цифровой сортировки.

6*. Применить DigitalSort для упорядочивания списка фамилий по возрастанию и убыванию.

7*. Экспериментально определить при каком количестве байтов в сортируемых числах цифровая сортировка DigitalSort начинает работать медленнее, чем QuickSort.

2.4. Задания по хешированию методом прямого связывания

1. Для набора из 12 первых последовательных неповторяющихся символов ФИО студента выполнить хеширование вручную методом прямого связывания (размер хеш-таблицы равен 5). При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Реализовать хеширование методом прямого связывания. Определить размер хеш-таблицы (количество списков) так, чтобы поиск был быстрее двоичного поиска. Предусмотреть подсчет фактического количества коллизий Кф. Вывести на экран построенные списки,

3. Исследовать зависимость количества коллизий от размера хеш-таблицы (в качестве размера хеш-таблицы взять десять простых чисел от 11 до 101), построить таблицу:

Размер хеш-таблицы	Количество исходных символов	Количество коллизий

4*. Организовать поиск элемента с заданным ключом для метода прямого связывания (для подтверждения выводить на экран номер списка и позицию искомого элемента в списке).

2.5. Задания по хешированию методом открытой адресации

1. 1. Для набора из 12 первых последовательных неповторяющихся символов ФИО студента выполнить хеширование вручную методом открытой адресации (линейные и квадратичные пробы, размер хеш-таблиц равен 11). При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Реализовать хеширование методом открытой адресации. Использовать два способа разрешения коллизий: линейные и квадратичные пробы. Предусмотреть подсчет фактического количества коллизий Кф. Вывести на экран заполненные хеш-таблицы и сравнить их.

Номер ячейки	0	1	2	3	…						…	m-1
Символ												

3.Подсчитать и сравнить количество коллизий при линейных и квадратичных пробах, в качестве размера хеш-таблицы взять десять простых чисел от 11 до 101. Построить таблицу:

Размер хеш-таблицы	Количество исходных символов	Количество коллизий	
		Линейные пробы	Квадратичные пробы

4*. Организовать поиск элемента с заданным ключом для метода открытой адресации при линейных и квадратичных пробах (для подтверждения выводить на экран номер позиции искомого элемента в хеш-таблицах)..