Задача 1

Полностью объяснить задачу из второго задания № 8. Под «полностью объяснить» подразумевается все — от свойств используемых структур данных до строчек кода.

Формат сдачи — предоставленный исходный код плюс устная беседа по нему.

Задача 2

Разобрать алгоритм интерполяционного поиска элемента в массиве (см. ниже). Оценить асимптотическую сложность алгоритма.

Формат сдачи — названная асимптотическая сложность (как «O большое от чего-то») плюс устная беседа по алгоритму.

Задача 3

На базе любой удобной задачи второго задания реализовать структуру данных и интерфейс к ней (указаны ниже). Программа при запуске должна читать команды из файла commands.txt, выполнять их, печатать результат на экран и завершаться. Никаких интерактивных действий от пользователя не предполагается.

Структура данных: стек на базе двухсвязного списка

Тип данных: строка не более 15 символов

Состав инструкций в commands.txt:

PUSH A — добавить элемент со значением A в вершину стека, ничего не печатать.

POP — забрать верхний элемент стека, элемент при этом из стека удаляется.

TEST A — проверить, равен ли верхний элемент стека A. Если равен — напечатать YES, если не равен — напечатать NO.

DEPTH — распечатать количество элементов в стеке.

Пример:

Инструкции в файле:

PUSH alice

PUSH bob

TEST alice

POP

DEPTH

Вывод программы:

NO (так как верхний элемент bob)

1 (так как элемент остался один)

Формат сдачи — предоставленный исходный код плюс устная беседа по нему.

Интерполяционный поиск элемента в массиве

Представьте себе, что Вы ищете слово в словаре. Маловероятно, что Вы сначала загляните в середину словаря, затем отступите от начала на 1/4 или 3/4 и т.д, как в бинарном поиске.

Если нужное слово начинается с буквы 'А', вы наверное начнете поиск где-то в начале словаря. Когда найдена отправная точка для поиска, ваши дальнейшие действия мало похожи на рассмотренные выше методы.

Если Вы заметите, что искомое слово должно находиться гораздо дальше открытой страницы, вы пропустите порядочное их количество, прежде чем сделать новую попытку. Это в корне отличается от предыдущих алгоритмов, не делающих разницы между 'много больше' и 'чуть больше'.

Мы приходим к алгоритму, называемому интерполяционным поиском: Если известно, что K лежит между Kl и Ku, то следующую пробу делаем на расстоянии (u-l)(K-Kl)/(Ku-Kl) от l, предполагая, что ключи являются числами, возрастающими приблизительно в арифметической прогрессии.

Пример реализации:

```
// Поиск в массиве K[1..N] числа X интерполяционным поиском
l=1; u=N;
while(u>=l) {
   i=l+(u-l)*(X-K[l])/(K[u]-K[l]);
   if(X<K[i]]  u=i-1;
   else if(X>K[i]] l=i+1;
   else НАШЛИ, X==K[i].
}
НЕ НАШЛИ
```