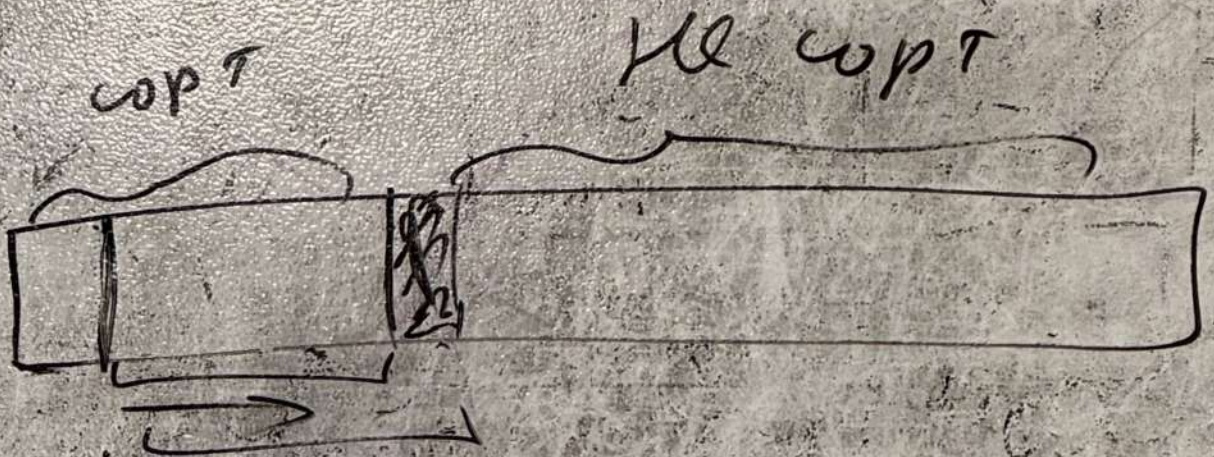
КП 9

1) ввести таблицу
2) сортируем / выводим
3) поиск в 5 чтк
до конца ввода:
- ввод ключа
- поиск по ключу

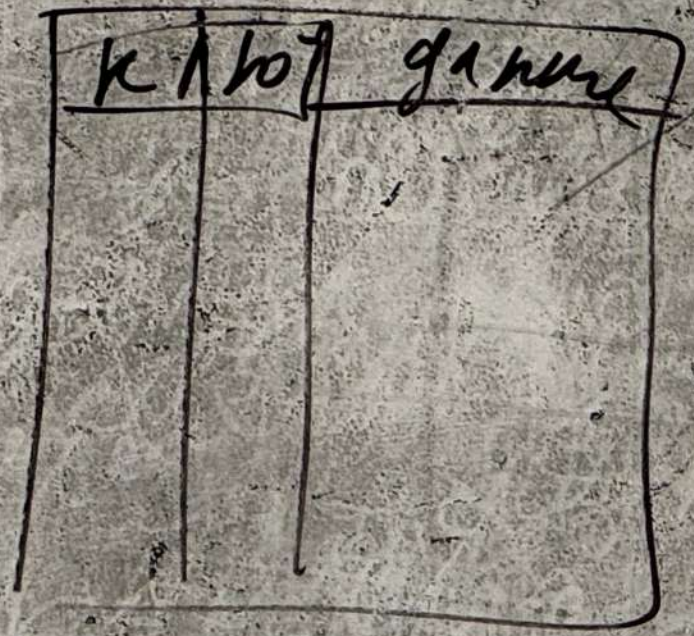


$$O(n * (\log n + n))$$

$$O(n^2)$$

```
struct key {
    double re;
    double im;
};

int compare( key* k1, key* k2) {
    if( k1->re > k2->re) return 1;
    else if( k1->re < k2->re ) return -1;
    else if( k1->im > k2->im){return 1;}
    else if( k1->im < k2->im)return -1;
    return 0;
}
```



int64_t

(long long)

сорт:

$(N * 7 + 5) \% 15 + 1$

структура:

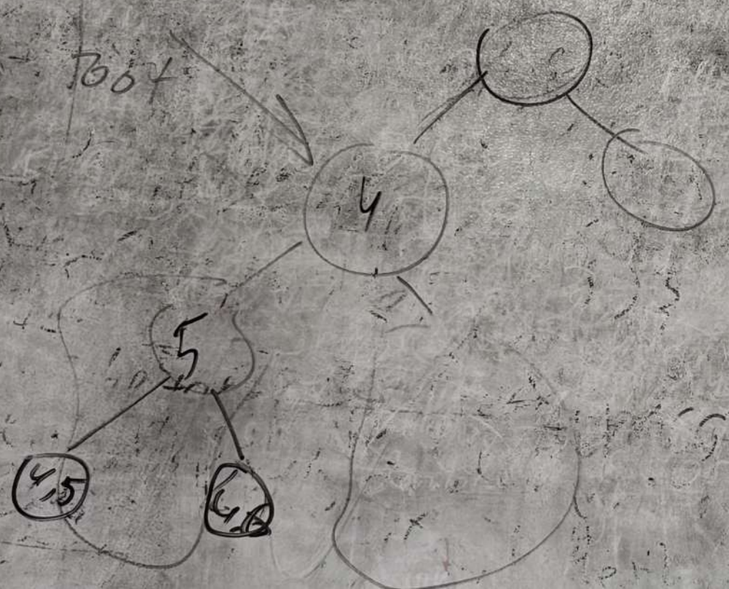
$(N * 3 + 2) \% 12 + 1$

```
void Leapify(elem *arr, int n){
    int parent = n/2 - 1;
    for(; parent >= 0; --parent)
        sift(arr, n, parent);
}

void Leap_sort(elem* arr, int n){
    Leapify(arr, n);
    for(; n>0; n--){
        swap(arr, 0, n-1);
        sift(arr, n-1, 0);
    }
}
```

КП 9

1) Ввести таблицу
2) сортировка / вывод
3) поиск в цикле до конца ввода:
   - ввод ключа
   - поиск по ключу



HEAP

```
void sift(elem *arr, int n, int root){
  int left = 2*root+1
  int right = 2*root+2;
  int maxx = root;
  if(left < n && compare(&arr[maxx].k, &arr[left].k) < 0){
      maxx = left;
  } // то же самое для right
  if(maxx == root) return;
  Swap(arr, maxx, root);
  // ↑ реализовать в самом
  sift(arr, n, maxx);
}
```

char[32]

1	7	1	5	6	0	1	20	9	9

для i-того эл-та:

родитель $= (i-1)/2$

левый $= 2 \cdot i + 1$

правый $= 2 \cdot i + 2$