CIS 435 Programming Exercise Module 6 (30points)

1 What to turn in

Follow the guidelines of Handout 10 dated January 23, 2004. Submissions that deviate from these guidelines will be assigned 0 points.

2 What to implement

Implementation is required for the function described in part A.

3 Part A: Implementation of Deterministic Select (20 points)

Provide an implementation of the Deterministic Selection algorithm with the following syntax and behavior.

```
void vselect(void *keys, int n, int stat, int size, int (*compare)( ), void
**re
sult);
```

`keys` is a pointer to the input array. Each element of the array is a datatype whose length in bytes is `size`. The length of the array (input size) is `n`. `compare` is a pointer to a function that returns an integer. Its two arguments are pointers to `void` as well. Depending on whether the first argument of `compare` is greater, equal, or less than the second, `compare` returns an 1, 0 or -1. Parameter `stat` is an integer between 1 (inclusive) and n (inclusive). The `stat`-th smallest of the n keys will be returned in `result` which holds the address of an array of size `size`.

Alternatively, your select may be defined as

```
int iselect(void *keys, int n, int stat, int size, int (*compare)( ));
```

In `iselect` the `stat`-th smallest keys of `keys` is not returned; its index in keys is however returned. This means the record containing the requested statistic begins at location $\&keys[iselect(keys, n, stat, size, mycompare) * size]$.

4 Part B: Experimental Results (10 points)

You will run your implementations with the testing functions provided in `sortg.c` on 2 different data-sets and 3 problem sizes.

1. **Problem sizes** are

   1. $n = 128000$.
   2. $n = 1024000$.
   3. $n = 4096000$.

2. The two different **data sets** consist of the following test instances.

   1. A reverse sorted array of integers where the i-th element of the array is $n - i$.

2. An array whose elements are randomly chosen using function `random` (see `sortg.c` on how to setup such an array).

a) For each problem size and test instance describe in tabular form (Table 1) the running time of your implementation. A timing of the execution of any function can be obtained similarly to the one provided in `sortg.c`.

5 Part C: Passing pointers

```
  main() {
  char *x;
  x=malloc(size*sizeof(char));

  iselect(keys,n,stat,size,compare,&x);
}

  iselect( ........, void **result)
{ char *statistic;

  statistic=malloc(size*sizeof(char));

  ...

  *result = (void *) statistic;
}
```

If you do this problem correctly, i.e. you collect all 30 points, you will receive 50 Bonus points for a total of 80 points from Problem 5 alone.