Computer Lab #2.

Revision of the C Programming Language

Task 1. Look through the given source code and analyze the *quicksort* program that uses a simple static array to store user data.

```c
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_MAX 5

// The compare function compares two values received as parameters.
// We have to specify this function with general parameter types, since
// being passed to the qsort, such approach insures the type safety.
// In other words, qsort does not care about types of elements being sorted.

int compare(const void* a, const void *b)
{
       // Returning value: <0, if a<b;
       //                   0, if a=b;
    //                 >0, if a>b.
       double* da=(double*)a, *db=(double*)b;
       if(*da<*db)return-1;
       else if (*da==*db) return 0;
       else return 1;
}

int main()
{
       double d[ARRAY_MAX];
       int i;
       for(i=0;i<ARRAY_MAX;i++)
       {
              printf("Enter a number:");
              scanf("%lf", &d[i]);
       }
       qsort(d,ARRAY_MAX,sizeof(double), compare);
       for(i=0;i<ARRAY_MAX;i++)
       {
              printf("%lf ",d[i]);
       }
       return 0;
}
```

Put a breakpoint inside the *compare* function, and check the values of pointer variables **da* and **db*. Is this code correct? Why do we use a function pointer? How can a function pointer be passed as a parameter? Why do the parameters of *compare* function have *void** types? Please recall from C, why the last parameter of *scanf* should be a pointer.

Task 2. Modify the given program so that the array size is not fixed anymore. Ask a user for the actual array size before reading the array elements. Consider dynamic memory management (recall it from C).

Task 3. Having the examples above, define the most important usage areas of pointers.

Task 4. Develop and implement a function that receives some string value as a parameter and reverses it. In the implementation you cannot use any special functions from external string libraries.