USER SIGN-IN

For this problem, the task is to implement the `user_sign_in` function. Assume this function is one of many endpoints in the backend of a larger web application.

```
void user_sign_in(char* identifier, char* password);
/*
    Input:
        identifier: string
        password: string

    Output:
        None
*/
```

The `user_sign_in` function accepts an identifier (which is either a username or an email address) and a password and handles the logic necessary to sign in a user.

How to sign in a user

- the user supplies the `identifier` and `password`
    - the `identifier` is either an email or a username.
    - the `get_identifier_type` helper function can be used to decide which is provided
- if the user-supplied credentials are correct:
    - set the user's logged in status to true
    - display the user's homepage
        - where user preferences are specified, those preferences have to be used in displaying the homepage
        - if the user has no preferences, use the default preferences specified at the top of the source code file, replicated below:

    ```
    #define DEFAULT_BANNER "blue"
    #define DEFAULT_FONT "arial"
    #define DEFAULT_TIMEOUT 60
    ```

- if the user-supplied credentials are incorrect:
    - display an error page to the user
- helper functions are provided for implementing the functionality above

Requirement

- the end product code should be **functional** and **secure**

Testing

You may uncomment the code in the `main` function to run a simple test of your implementation when you are done. The test uses the stub implementation to randomly test failed and successful sign in attempts. Assuming your code implements the correct functionality, you should see of one of the following outputs at random:

sucessful sign-in

```
SIGN IN FOR user_identifier SUCCESSFUL - HOMEPAGE
```

unsuccessful sign-in

```
SIGN IN FOR user_identifier NOT SUCCESSFUL, TRY AGAIN
```

Structs defined and used in the code

The `preferences` struct stores a user's preferences which are used to display the homepage

```
preferences
    {
        banner_color: string
        display_font: string
        timeout: int
    }
```

The `user` struct stores information about a user as well as their preferences.

```
user
    {
        username: string
        email: string
        phone_number: string
        age: int
        user_preferences: pointer to a `preferences` struct
    }
```

Helper Functions

Stub implementations for the helper functions have been provided in the source code file. You can assume that they function as they need to.

`get_identifier_type(identifier)`

```
char* get_identifier_type(char* identifier)
    /*
    Input:
        identifier: string

    Output:
        string

    Returns the identifier type used by the user.
    The type is either "username" or "email".
    */
```

`display_homepage(identifier)`

```
void display_homepage(char* identifier, char* banner_color, char*
display_font, int timeout)
    /*
    Input:
        identifier: string
        banner_color: string
        display_font: string
        timeout: int

    Output:
        None

    Sends a HTML response back to the client that renders their
    homepage after a successful sign-in attempt.
    */
```

`valid_user_credentials(identifier, password, is_username)`

```
user* valid_user_credentials(char* identifier, char* password, bool,
is_username)
    /*
    Input:
        identifier: string
        password: string
        is_username: boolean

    Output:
        user* (pointer to user struct)

    Returns a pointer to a user struct if a valid user is found or NULL
    if there is no valid user with the provided identifier/password.

    The 'is_username' flag is used to indicate whether the user
    signed in with a username or email address.
    */
```

display_error_page(identifier)

```
void display_error_page(char* identifier)
    /*
    Input:
        identifier: string

    Output:
        None

    Sends a HTML response back to the client that renders an error
    page upon an unsuccessful sign-in attempt.
    */
```

set_logged_in_status(identifier, logged_in)

```
void set_logged_in_status(char* identifier, bool logged_in)
    /*
    Input:
        identifier: string
        logged_in: boolean

    Output:
        None

    Sets user logged in cookies as well as other variables that need
    to be updated upon sign in.
    */
```

validate_email_format(email)

```
char* validate_email_format(char* email)
    /*
    Input:
        email: string

    Output:
        string

    Returns the email if it is syntactically valid or an empty string
    */
```

sanitize_input(char* input)

```
char* sanitize_input(char* input)
    /*
    Input:
        input: string

    Output:
        string

    Returns a sanitized version of the input to prevent against general
    kinds of attacks.
    */
```