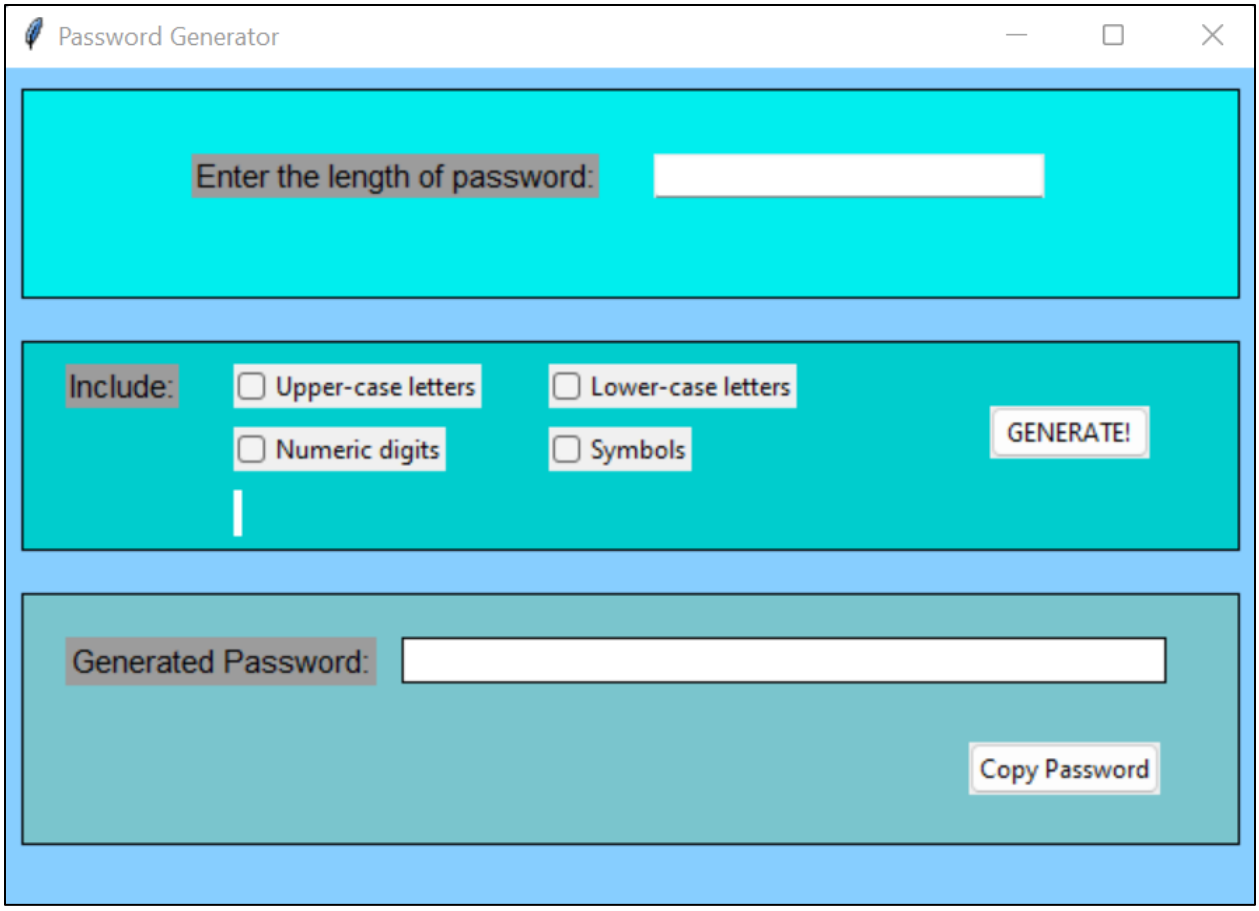
```python
import tkinter as tk
from tkinter import ttk
import string
import random

# Initialize the main application window
root = tk.Tk()
root.title("Password Generator")  # Set the title of the window
root.geometry('600x400')  # Set the window size
root.configure(bg="SkyBlue1")  # Set the background color of the window
large_font = ("Arial", 11)  # Define a common font for labels

# Create a frame for the input section
input_frame = tk.Frame(root, bg="cyan2", width=580, height=100, relief="solid", bd=1)
input_frame.place(x=10, y=10)

# Create a frame for the checkboxes section
checkbox_frame = tk.Frame(root, bg="cyan3", width=580, height=100, relief="solid", bd=1)
checkbox_frame.place(x=10, y=130)

# Create a frame for the output section
output_frame = tk.Frame(root, bg="CadetBlue3", width=580, height=120, relief="solid", bd=1)
output_frame.place(x=10, y=250)

# Input length label
length_label = ttk.Label(input_frame, text="Enter the length of password:", font=large_font, background= "grey61")
length_label.place(x=80, y=30)

# Dropdown for selecting password length
password_length = ttk.Entry(input_frame, justify="center", width=30, background="white")
password_length.place(x=300, y=30)

# Warning label for invalid selections
warning_label = ttk.Label(input_frame, text="", font=("Arial", 11), foreground="red4", background= "cyan2")
warning_label.place(x=300, y=60)

# Function to display a warning when no length is selected
```

```
def on_combo_select(event):
    if not password_length.get():  # Check if no value is selected
        warning_label.config(text="Please select the length of password characters.")
    else:
        warning_label.config(text="")  # Clear the warning when valid

# Bind the Combobox selection event to the function
password_length.bind("<<ComboboxSelected>>", on_combo_select)

# Checkboxes for selecting character types
includes = ttk.Label(checkbox_frame, text='Include:', font=large_font, background= "grey61")
includes.place(x=20, y=10)

# Variables to track checkbox states
var1 = tk.IntVar()  # Upper-case letters
var2 = tk.IntVar()  # Lower-case letters
var3 = tk.IntVar()  # Numeric digits
var4 = tk.IntVar()  # Symbols

# Add checkboxes for each character type
checkbox1 = ttk.Checkbutton(checkbox_frame, text='Upper-case letters', variable=var1, command= lambda: calculate_complexity())
checkbox1.place(x=100, y=10)
checkbox2 = ttk.Checkbutton(checkbox_frame, text='Lower-case letters', variable=var2, command= lambda: calculate_complexity())
checkbox2.place(x=250, y=10)
checkbox3 = ttk.Checkbutton(checkbox_frame, text='Numeric digits', variable=var3, command= lambda: calculate_complexity())
checkbox3.place(x=100, y=40)
checkbox4 = ttk.Checkbutton(checkbox_frame, text='Symbols', variable=var4, command= lambda: calculate_complexity())
checkbox4.place(x=250, y=40)

# Output frame labels and password display entry
generated_password_label = tk.Label(output_frame, text="Generated Password:", background= "grey61", font=("Arial", 11))
generated_password_label.place(x=20, y=20)

# Entry box to display the generated password
password_entry = tk.Entry(output_frame, width=40, font=("Arial", 12), justify="center", relief="solid")
password_entry.place(x=180, y=20)

```

```python
# Add a label for displaying password complexity
complexity_label = ttk.Label(checkbox_frame, text="", font=("Arial", 12), background="white")
complexity_label.place(x=100, y=70)

# Function to calculate and display password complexity
def calculate_complexity():
    selected_count = sum([var1.get(), var2.get(), var3.get(), var4.get()])  # Count selected options
    if selected_count == 1:
        complexity_label.config(text="Complexity: Very Weak", foreground="red", background= "white")
    elif selected_count == 2:
        complexity_label.config(text="Complexity: Weak", foreground="orange", background= "white")
    elif selected_count == 3:
        complexity_label.config(text="Complexity: Strong", foreground="blue", background= "white")
    elif selected_count == 4:
        complexity_label.config(text="Complexity: Very Strong", foreground="green", background= "white")
    else:
        complexity_label.config(text="Complexity: Select at least one option", foreground="gray21", background= "white")

# Label to display a warning when no character types are selected
not_selected_options = ttk.Label(checkbox_frame, text='', background="white", foreground="gray21")
not_selected_options.place(x=100, y=70)

# Function to generate the password
def final_generation():
    if not password_length.get():  # Check if password length is not selected
        warning_label.config(text="Please select the password length first.")
        return

    getting_length = int(password_length.get())  # Get the selected length
    selected_characters = ""

    # Add characters based on selected options
    if var1.get():
        selected_characters += string.ascii_uppercase
    if var2.get():
        selected_characters += string.ascii_lowercase
    if var3.get():
```

```
        selected_characters += string.digits
    if var4.get():
        selected_characters += string.punctuation

    # Generate password if at least one option is selected
    if selected_characters:
        result = random.choices(selected_characters, k=getting_length)  # Randomly select characters
        password_entry.delete(0, tk.END)  # Clear the entry box
        password_entry.insert(0, "".join(result))  # Insert the generated password
    else:
        complexity_label.config(text="Select at least one character type.", font=("Arial", 11))
        password_entry.delete(0, tk.END)

# Button to generate the password
final_button = ttk.Button(checkbox_frame, text='GENERATE!', command=final_generation)
final_button.place(x=460, y=30)

# Function to copy password to clipboard
def copy_to_clipboard():
    root.clipboard_clear()  # Clear the clipboard
    root.clipboard_append(password_entry.get())  # Append the password to the clipboard
    root.update()  # Update the clipboard

# Button to copy the generated password
copy_button = ttk.Button(output_frame, text="Copy Password", command=copy_to_clipboard)
copy_button.place(x=450, y=70)

# Run the application
root.mainloop()
```

Output:


Password Generator
Enter the length of password:
Include:
Upper-case letters
Lower-case letters
Numeric digits
Symbols
GENERATE!
Generated Password:
Copy Password