```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def celsius_to_kelvin(celsius):
    return celsius + 273.15

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def fahrenheit_to_kelvin(fahrenheit):
    return (fahrenheit - 32) * 5/9 + 273.15

def kelvin_to_celsius(kelvin):
    return kelvin - 273.15

def kelvin_to_fahrenheit(kelvin):
    return (kelvin - 273.15) * 9/5 + 32

def convert_temperature(value, unit):
    if unit == "C":
        celsius = value
        fahrenheit = celsius_to_fahrenheit(celsius)
        kelvin = celsius_to_kelvin(celsius)
    elif unit == "F":
        fahrenheit = value
        celsius = fahrenheit_to_celsius(fahrenheit)
        kelvin = fahrenheit_to_kelvin(fahrenheit)
    elif unit == "K":
        kelvin = value
        celsius = kelvin_to_celsius(kelvin)
        fahrenheit = kelvin_to_fahrenheit(kelvin)
    else:
        return "Invalid unit"

    return celsius, fahrenheit, kelvin

def main():
    value = float(input("Enter the temperature value: "))
    unit = input("Enter the unit of the temperature (C for Celsius, F for Fahrenheit, K for Kelvin): ").upper()

    result = convert_temperature(value, unit)

    if result != "Invalid unit":
        celsius, fahrenheit, kelvin = result
        print(f"Temperature in Celsius: {celsius:.2f}°C")
        print(f"Temperature in Fahrenheit: {fahrenheit:.2f}°F")
        print(f"Temperature in Kelvin: {kelvin:.2f}K")
    else:
        print("Invalid unit entered. Please enter C, F, or K.")
```

```
if __name__ == "__main__":
    main()
```

```
Enter the temperature value: 70
Enter the unit of the temperature (C for Celsius, F for Fahrenheit, K for Kelvin): 5
Invalid unit entered. Please enter C, F, or K.
```

```
contacts = []

def add_contact():
    name = input("Enter contact name: ")
    phone = input("Enter phone number: ")
    email = input("Enter email address: ")
    contact = {"name": name, "phone": phone, "email": email}
    contacts.append(contact)
    print("Contact added successfully.")

def view_contacts():
    if not contacts:
        print("No contacts found.")
    else:
        print("Contact List:")
        for contact in contacts:
            print(f"Name: {contact['name']}")
            print(f"Phone: {contact['phone']}")
            print(f"Email: {contact['email']}")
            print("---")

def edit_contact():
    name = input("Enter the name of the contact to edit: ")
    for contact in contacts:
        if contact["name"] == name:
            new_phone = input(f"Enter new phone number (current: {contact['phone']}): ")
            new_email = input(f"Enter new email address (current: {contact['email']}): ")
            contact["phone"] = new_phone
            contact["email"] = new_email
            print("Contact updated successfully.")
            return
    print("Contact not found.")

def delete_contact():
    name = input("Enter the name of the contact to delete: ")
    for contact in contacts:
        if contact["name"] == name:
            contacts.remove(contact)
            print("Contact deleted successfully.")
            return
    print("Contact not found.")

while True:
    print("\nContact Manager")
    print("1. Add Contact")
    print("2. View Contacts")
    print("3. Edit Contact")
    print("4. Delete Contact")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")
```

```
    if choice == "1":
        add_contact()
    elif choice == "2":
        view_contacts()
    elif choice == "3":
        edit_contact()
    elif choice == "4":
        delete_contact()
    elif choice == "5":
        print("Exiting Contact Manager...")
        break
    else:
        print("Invalid choice. Please try again.")
```

```
...
 Contact Manager
 1. Add Contact
 2. View Contacts
 3. Edit Contact
 4. Delete Contact
 5. Exit
 Enter your choice (1-5): 1
 Enter contact name: 6
 Enter phone number: 3
 Enter email address: shivani@gmail.com
 Contact added successfully.

 Contact Manager
 1. Add Contact
 2. View Contacts
 3. Edit Contact
 4. Delete Contact
 5. Exit
 Enter your choice (1-5):
```

```
import random

def guess_the_number():
    # Generate a random number between 1 and 100
    number_to_guess = random.randint(1, 100)
    attempts = 0
    guessed_correctly = False

    print("Welcome to the Guess the Number Game!")
    print("I have generated a random number between 1 and 100.")
    print("Try to guess it!")

    while not guessed_correctly:
        try:
            # Prompt the user for a guess
            user_guess = int(input("Enter your guess: "))
            attempts += 1

            # Compare the guess to the generated number
            if user_guess < number_to_guess:
                print("Your guess is too low. Try again!")
            elif user_guess > number_to_guess:
                print("Your guess is too high. Try again!")
            else:
                guessed_correctly = True
                print(f"Congratulations! You've guessed the number {number_to_guess} in {attempts} attempts.")
        except ValueError:
            print("Please enter a valid integer.")

if __name__ == "__main__":
    guess_the_number()


def print_grid(grid):
    """Function to print the Sudoku grid."""
    for row in grid:
        print(" ".join(str(num) if num != 0 else '.' for num in row))

def find_empty_location(grid):
    """Function to find an empty location in the grid (represented by 0)."""
    for i in range(9):
        for j in range(9):
            if grid[i][j] == 0:
                return (i, j)  # row, col
    return None

def is_safe(grid, row, col, num):
    """Function to check if it's safe to place a number in the given location."""
    # Check the row
    if num in grid[row]:
        return False
```

```
        # Check the column
        for i in range(9):
            if grid[i][col] == num:
                return False

        # Check the 3x3 box
        box_row_start = row - row % 3
        box_col_start = col - col % 3
        for i in range(3):
            for j in range(3):
                if grid[i + box_row_start][j + box_col_start] == num:
                    return False

        return True

    def solve_sudoku(grid):
        """Function to solve the Sudoku puzzle using backtracking."""
        empty_loc = find_empty_location(grid)
        if not empty_loc:
            return True  # Puzzle solved

        row, col = empty_loc
```