LAB # 01

Spring 2024

Data Structures and Algorithm

Submitted by: **Hassan Zaib Jadoon**

Registration No.: **22PWCSE2144**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Usman Mailk

February 18, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Problem 1:

a) Write a recursive implementation of Factorial.

b) Write an iterative (i.e. use any of the looping constructs (for, do-while, while) to accomplish the task) implementation of Factorial.

Solution:

Part a:

Code:

Start here | Lab 01 Tasks 1 a.cpp

```cpp
//Write a recursive implementation of Factorial.

#include <iostream>
using namespace std;
long long factorial(int n) {
  if (n == 0 || n == 1) {
    return 1;}
     else {
    return n * factorial(n - 1);
            }
                }
int main() {
  int num;

   cout << "Enter a Positive integer: ";
   cin >> num;

  if (num < 0) {
     cout << "Error: Factorial is not defined for given numbers." <<  endl;
  } else {
    long long result = factorial(num);
     cout << "The factorial of " << num << " is " << result <<  endl;
  }

  return 0;
}
```

Output:

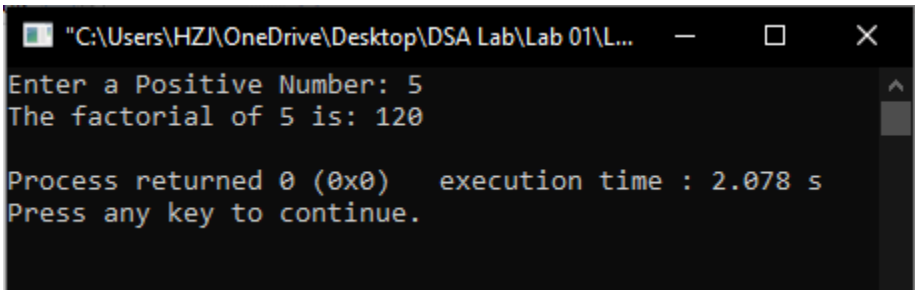
```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\Lab 01 Tasks 1 a.exe"
Enter a Positive integer: 4
The factorial of 4 is 24

Process returned 0 (0x0)   execution time : 5.547 s
Press any key to continue.
```

Part b:

Code:

```cpp
// b)   Write an iterative (i.e. use any of the looping constructs (for, do-while, while) to accomplish the task) implementation of factorial
#include <iostream>

using namespace std;

int factorial(int n) {
  if (n == 0 || n == 1) {
    return 1;
  }
  int fact = 1;

  for (int i = 2; i <= n; ++i) {
    fact *= i;
  }

  return fact;
}

int main() {
  int number;
  cout << "Enter a Positive Number: ";
  cin >> number;

  int result = factorial(number);
  cout << "The factorial of " << number << " is: " << result << endl;

  return 0;
}
```

Output:



```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\L...
Enter a Positive Number: 5
The factorial of 5 is: 120

Process returned 0 (0x0)   execution time : 2.078 s
Press any key to continue.
```

Problem 2:

a) Write a recursive implementation of Fibonacci.
b) Write an iterative implementation of Fibonacci.

Solution:

Part a:

Code:

```cpp
// a)   Write a recursive implementation of Fibonacci.
#include <iostream>
using namespace std;

void fibonacci(int n, int current = 0, int previous = 1) {
    // Base case
    if (n == 0) {
        cout << current << " ";
        return;
    }

    // Print the current number and call recursively
    cout << current << " ";
    fibonacci(n - 1, previous, current + previous);
}

int main() {
    int number;
    cout << "Enter a non-negative integer: ";
    cin >> number;

    fibonacci(number);
    cout << endl;

    return 0;
}
```

Output:

```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\Lab 01 Task 2 a.exe"
Enter a non-negative integer: 4
0 1 1 2 3

Process returned 0 (0x0)   execution time : 2.109 s
Press any key to continue.
```

Part b:

Code:

Start here | Lab 01 Tasks 1 a.cpp | Lab 01 Task 1 b.cpp | Lab 01 Task 2 a.cpp

```cpp
// a)   Write a recursive implementation of Fibonacci.
#include <iostream>

using namespace std;

int factorial(int n) {
  if (n == 0 || n == 1) {
    return 1;
  }
  int fact = 1;

  for (int i = 2; i <= n; ++i) {
    fact *= i;
  }

  return fact;
}

int main() {
  int number;
  cout << "Enter a Positive Number: ";
  cin >> number;

  int result = factorial(number);
  cout << "The factorial of " << number << " is: " << result << endl;

  return 0;
}
```

Output:

```
"C:\Users\HZ\OneDrive\Desktop\DSA Lab\Lab 01\Lab 01 Task 2 a.exe"
Enter a non-negative integer: 5
0 1 1 2 3 5

Process returned 0 (0x0)   execution time : 2.172 s
Press any key to continue.
```

Problem 3:

The **greatest common divisor** (g. c. d.) of two nonnegative integers is the largest integer that divides evenly into both. In the third century B.C., the Greek mathematician Euclid discovered that the greatest common divisor of *x* and *y* can always be computed as follows:

- If *x* is evenly divisible by *y*, then *y* is the greatest common divisor.
- Otherwise, the greatest common divisor of *x* and *y* is always equal to the greatest common divisor of *y* and the remainder of *x* divided by *y*. Use Euclid's insight to write a recursive function **GCD (x, y)** that computes the greatest common divisor of *x* and *y*.

Solution:

Code:

```
Start here | Lab 01 Tasks 1 a.cpp | Lab 01 Task 1 b.cpp | Lab 01 Task 2 a.cpp | *P3.cpp

/* The greatest common divisor (g.c.d.) of two nonnegative integers is the largest integer that divides evenly into both.
In the third century B.C., the Greek mathematician Euclid discovered that the greatest common divisor of x and y can always be computed as follows:
If x is evenly divisible by y, then y is the greatest common divisor.
Otherwise, the greatest common divisor of x and y is always equal to the greatest common divisor of y and the remainder of x divided by y.
Use Euclid's insight to write a recursive function GCD(x,y) that computes the greatest common divisor of x and y*/

#include <iostream>

using namespace std;
int GCD(int x, int y) {
    // Base case
    if (y == 0) {
        return x; // When y becomes 0, x is the GCD
    } else {
        // Recursive case
        return GCD(y, x % y); // Call GCD with y and remainder of x/y
    }
}

int main() {
    int x, y;
    cout << "Enter two non-negative integers: ";
    cin >> x >> y;

    if (x < 0 || y < 0) {
        cout << "Error: Please enter non-negative integers." << endl;
    } else {
        int gcd = GCD(x, y);
        cout << "The greatest common divisor of " << x << " and " << y << " is: " << gcd << endl;
    }

    return 0;
}
```

Output:

```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\P3.exe"
Enter two non-negative integers: 4
2
The greatest common divisor of 4 and 2 is: 2

Process returned 0 (0x0)   execution time : 5.281 s
Press any key to continue.
```

Problem 4:

Unlike many programming languages, C++ does not include a predefined operator that raises a number to a power. As a partial remedy for this deficiency, write a recursive implementation of a function int **RaiseIntToPower(int n, int k)** that calculates n^k. The recursive insight that you need to solve this problem is the mathematical property that

$$n^k = \begin{cases} 1 & \text{if } k = 0 \\ n \times n^{k-1} & \text{otherwise} \end{cases}$$

Solution:
Code:

Start here | Lab 01 Tasks 1 a.cpp | Lab 01 Task 1 b.cpp | Lab 01 Task 2 a.cpp | *P3.cpp | P4.cpp

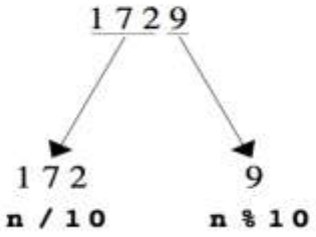
```cpp
#include <iostream>
using namespace std;
int RaiseIntToPower(int n, int k) {

  if (k == 0) {
    return 1;
  }
  if (k < 0) {
    return 1 / RaiseIntToPower(n, -k);
  }

  int halfK = k / 2;
  int halfPower = RaiseIntToPower(n, halfK);
  if (k % 2 == 0) {
    return halfPower * halfPower;
  } else {
    return halfPower * halfPower * n;
  }
}

int main() {
  int base, exponent;
  cout << "Enter a base number: ";
  cin >> base;
  cout << "Enter an exponent: ";
  cin >> exponent;

  int result = RaiseIntToPower(base, exponent);
  cout << base << " raised to the power of " << exponent << " is: " << result << endl;

  return 0;
}
```

Output:

```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\P4.exe"
Enter a base number: 2
Enter an exponent: 4
2 raised to the power of 4 is: 16

Process returned 0 (0x0)   execution time : 3.297 s
Press any key to continue.
```

Problem 5:

Write a recursive function **DigitSum(n)** that takes a nonnegative integer and returns the sum of its digits. For example, calling **DigitSum(1729)** should return 1 + 7 + 2 + 9, which is 19. The recursive implementation of **DigitSum** depends on the fact that it is very easy to break an integer down into two components using division by 10. For example, given the integer 1729, you can divide it into two pieces as follows:



Each of the resulting integers is strictly smaller than the original and thus represents a simpler case.

Solution:

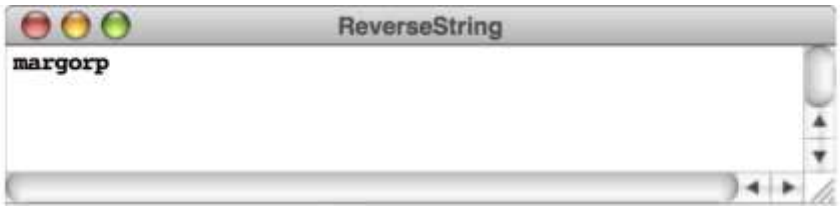
```
#include <iostream>
int DigitSum(int n) {
  // Base case: single-digit number
  if (n < 10) {
    return n;
  }

  // Recursive case: sum the last digit and the sum of remaining digits
  return n % 10 + DigitSum(n / 10);
}

int main() {
  int number;
  std::cout << "Enter a non-negative integer: ";
  std::cin >> number;

  int sum = DigitSum(number);
  std::cout << "The sum of digits for " << number << " is: " << sum << std::endl;

  return 0;
}
```

Output:

```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\P5.exe"
Enter a non-negative integer: 421
The sum of digits for 421 is: 7

Process returned 0 (0x0)   execution time : 2.125 s
Press any key to continue.
```

Problem 6:

Write a recursive function that takes a string as argument and returns the reverse of that string. The prototype for this function should be **string Reverse(string str);** and the statement
cout << Reverse("program") << endl; should display



Your solution should be entirely recursive and should not use any iterative constructs such as **while** or **for**.

Solution:

Code:

```
#include <iostream>
#include <string>

using namespace std;

string Reverse(string str) {
  // Base case: empty string or single character
  if (str.length() <= 1) {
    return str;
  }

  // Recursive case: reverse substring excluding first character
  string reversedRest = Reverse(str.substr(1));

  // Append first character to the reversed rest
  return reversedRest + str[0];
}

int main() {
  string original = "program";
  string reversed = Reverse(original);

  cout << "Original string: " << original << endl;
  cout << "Reversed string: " << reversed << endl;

  return 0;
}
```

Output:

```
"C:\Users\HZJ\OneDrive\Desktop\DSA Lab\Lab 01\p6.exe"
Original string: program
Reversed string: margorp

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```