**Rajshahi University of Engineering & Technology**

CSE 2102: Sessional Based on CSE 2101

Lab Report 08

Dated: 27.06.18

Submitted to

Rizoan Toufiq

Instructor, CSE 2102

Assistant Professor

Dept. of Computer Science & Engineering

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of Computer Science & Engineering

**Experiment No. 3**

**Name of the Experiment: Algorithms, Number Theory and Cryptography.**

**1. EXPERIMENT [11]**

Given a set of identification numbers, use a hash function to assign them to

memory locations where there are k memory locations.

**SOLUTION:**

#include <iostream>

#include <cmath>

using namespace std;

int hassh[10];

void check() {

for(int i = 0; i < 10; i++) {

for(int j = 0; j < i; j++) {

if(hassh[i] == hassh[j]) {

hassh[i]++;

}

}

}

}

int main() {

int id[10];

int k;

cout << "K = ";

cin >> k;

cout << "Id(s) = ";

for(int i = 0; i < 10; i++) {

cin >> id[i];

}

for(int i = 0; i < 10; i++) {

hassh[i] = id[i] % k;

}

for(int i = 0; i < 10; i++) {

for(int j = 0; j < i; j++) {

if(hassh[i] == hassh[j]) {

hassh[i]++;

}

}

}

check();

check();

check();

cout << "Index position:" << endl;

for(int i = 0; i < 10; i++) {

cout << id[i] << " -> " << hassh[i] << endl;

}

}

OUTPUT:

K = 8

Id(s) = 3 6 3 6 4 2 5 8 9 4

Index position:

3 -> 3

6 -> 6

3 -> 4

6 -> 7

4 -> 5

2 -> 2

5 -> 8

8 -> 0

9 -> 1

4 -> 9

**Discussion:** Here, Hash Function is implemented logically and it is checked if there is threre overlapping indexical position occur to the ids.

**2. EXPERIMENT [12]**

Given a positive integer N, a modulus m, multiplier a, increment c, and seed

x0, where 0 ≤ a<m, 0 ≤ c<m, and 0 ≤ x0 <m, generate the sequence of N

pseudo-random numbers using the linear congruential generator xn+1 = (axn +

c) mod m.

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

int n, m, a, c, xo, i, new\_x;

cout << "Enter N, m, a, c, xo (respectively): ";

cin >> n >> m >> a >> c >> xo;

cout << "Generated Psudo-random numbers are: ";

for(i = 0; i < n; i++) {

new\_x = (a\*xo + c) % m;

cout << new\_x << " ";

xo = new\_x;

}

}

OUTPUT:

Enter N, m, a, c, xo (respectively): 10 71 12 54 39

Generated Psudo-random numbers are: 25 70 42 61 5 43 2 7 67 6

**3. EXPERIMENT [24]**

Given the positive integers a, b, and m with m> 1, find ab mod m.

**SOLUTION:**

#include <iostream>

#include <cmath>

using namespace std;

unsigned long long a[100];

unsigned long long counter = 0;

void binary (unsigned long long n) {

unsigned long long b = 2;

unsigned long long q = n, k = 0;

while(q) {

a[k] = q % b;

q = floor(q/b);

k++;

counter++;

}

k--;

}

int main() {

unsigned long long b, x, power, m, i, n;

x = 1;

cout << "Enter a, b, m (respectively): ";

cin >> b >> n >> m;

binary(n);

power = b % m;

for(i = 0; i < counter; i++) {

if(a[i] == 1) {

x = (x \* power) % m;

}

power = (power \* power) % m;

}

cout << "\n" << b << "^" << n << " mod " << m << " = " << x << endl;

}

OUTPUT:

Enter a, b, m (respectively): 7 5 14

7^5 mod 14 = 7

**4. EXPERIMENT [26]**

Given two positive number a and b, find s and t such that gcd(a,b) = sa+tb.

**SOLUTION:**

#include <iostream>

using namespace std;

int gcd (int a, int b) {

int r;

while(b) {

r = a % b;

a = b;

b = r;

}

return a;

}

int main() {

int a, b, temp;

cin >> a >> b;

if(a > b) {

temp = a;

a = b;

b = temp;

}

int gcd\_value = gcd(a, b);

if (1 == gcd\_value) {

cout << "GCD: " << gcd\_value << " = ";

cout << a << " \* " << 0 << " + " << b << " \* " << 0 << endl;

return 0;

}

for(int i = 0; i <= gcd\_value; i++) {

for(int j = 0; j <= gcd\_value; j++) {

if(a \* i + b \* j == gcd\_value) {

cout << "GCD: " << gcd\_value << " = ";

cout << a << " \* " << i << " + " << b << " \* " << j << endl;

}

}

}

}

OUTPUT:

18 6

GCD: 6 = 6 \* 1 + 18 \* 0

**Experiment No. 4**

**Name of the Experiment: Induction and Recursion**

**1. EXPERIMENT [4]**

Given a string, find its reversal.

**SOLUTION:**

#include <iostream>

#include <string.h>

using namespace std;

int main() {

string s;

cout << "Input String: ";

getline(cin, s);

cout << "Reversed String: ";

for(int i = s.length() - 1; i >= 0; i--) {

cout << s[i];

}

return 0;

}

OUTPUT:

Input String: Fuad Al Abir

Reversed String: ribA lA dauF

**2. EXPERIMENT [5]**

Given a real number a and a non negative integer n, find an using recursion.

**3. EXPERIMENT [06]**

Given a real number a and a non-negative integer n, find (a2)^ n using recursion.

**SOLUTION:**

#include <iostream>

using namespace std;

unsigned long long power(unsigned long long a, unsigned long long n) {

if(n == 0) return 1;

else if(n == 1) return a;

else if(n > 1) return a\*power(a, n - 1);

}

int main()

{

unsigned long long a, n;

cin >> a >> n;

unsigned long long result = power(a, power(2, n));

cout << a << " ^ " << 2 << " ^ " << n << " = ";

cout << result;

return 0;

}

OUTPUT:

4 4

4 ^ 2 ^ 4 = 4294967296

**4. EXPERIMENT [08]**

Given two integers not both zero, find their greatest common divisor using recursion.

**SOLUTION:**

#include <iostream>

using namespace std;

int euclidian\_gdc(int a, int b) {

int r;

if(b == 0) return a;

else {

r = a % b;

a = b;

b = r;

euclidian\_gdc(a, b);

}

}

int main()

{

int a, b;

cin >> a >> b;

cout << "GCD: " << euclidian\_gdc(a, b);

return 0;

}

OUTPUT:

18 14

2

18 6

6

**5. EXPERIMENT [09]**

Given a list of integers and an element x, locate x in this list using a recursive implementation of a linear search.

**SOLUTION:**

#include <iostream>

using namespace std;

int linear\_search(int arr[], int n, int siz, int pos) {

if(siz < pos) return -1;

else if(arr[pos] == n) return pos;

else return linear\_search(arr, n, siz, pos+1);

}

int main()

{

int arr[10];

int siz = sizeof(arr)/sizeof(arr[0]);

cout << "Enter 10 numbers: ";

for(int i = 0; i < siz; i++) {

cin >> arr[i];

}

cout << "Number to be searched: ";

int n;

cin >> n;

cout << "Index of the number " << n << "is: ";

cout << linear\_search(arr, n, siz, 0);

return 0;

}

OUTPUT:

Enter 10 numbers: 2 4 6 3 12 1 16 9 7 0

Number to be searched: 1

Index of the number 1is: 5

**6. EXPERIMENT [11]**

Given a non negative integer n, find the nth Fibonacci number using iteration.

**SOLUTION:**

#include <iostream>

using namespace std;

int main()

{

int n;

cin >> n;

int a = 0, b = 1, count = 2;

cout << a << " " << b << " ";

while(count != n) {

int temp = b;

b = a + b;

a = temp;

count++;

cout << b << " ";

}

return 0;

}

OUTPUT:

13

0 1 1 2 3 5 8 13 21 34 55 89 144

**7. EXPERIMENT [12]**

Given a nonnegative integer n, find the nth Fibonacci number using recursion.

**SOLUTION:**

#include <iostream>

using namespace std;

int fibonacci(int n, int a, int b, int count) {

if(n == 1) return 0;

else if(count >= n) return b;

else {

int temp = b;

b = a + b;

a = temp;

cout << a << " ";

return fibonacci(n, a, b, ++count);

}

}

int main()

{

int n;

cin >> n;

cout << "0 ";

cout << fibonacci(n, 0, 1, 2);

return 0;

}

OUTPUT:

13

0 1 1 2 3 5 8 13 21 34 55 89 144

**Experiment No. 5**

**Name of the Experiment: Induction and Recursion**

**1. EXPERIMENT [01]**

Given a positive integer n and a non negative integer not exceeding n, find the

number of r-permutations and r-combinations of a set with n elements.

**SOLUTION:**

#include <iostream>

using namespace std;

int fact(int n) {

if(n <= 1) return 1;

else return n\*fact(n - 1);

}

int main()

{

int n, r;

cin >> n >> r;

cout << "Number of permutations: " << fact(n) / fact(n - r) << endl;

cout << "Number of combinations: " << fact(n) / fact(n - r) / fact(r) << endl;

return 0;

}

OUTPUT:

5 3

Number of permutations: 60

Number of combinations: 10

**1. EXPERIMENT [01]**

Given positive integers n and r, find the number of r-permutations when repetition is allowed and r-combinations when repetition is allowed of a set with n elements.

**SOLUTION:**

#include <iostream>

using namespace std;

int fact(int n) {

if(n <= 1) return 1;

else return n\*fact(n - 1);

}

int power(int n, int r) {

if(r == 0) return 1;

else if(r == 1) return n;

else return n\*power(n, --r);

}

int main()

{

int n, r;

cin >> n >> r;

cout << "Number of permutations (with repetitions): "

<< power(n, r) << endl;

cout << "Number of combinations (with repetitions): "

<< fact(n + r - 1) / fact(n - 1) / fact(r) << endl;

return 0;

}

OUTPUT:

5 3

Number of permutations (with repetitions): 125

Number of combinations (with repetitions): 35