1.

STACK IMPLEMENTATION USING LIST

stack = ["SOBAH", "SIMARA", "ARONY"]

stack.append("RIHAN")

stack.append("ISHAO")

print(stack)

# Removes the last item

print(stack.pop())

print(stack)

# Removes the last item

print(stack.pop())

print(stack)

OUTPUT:

============== RESTART: C:/Users/abc/Desktop/pthon clg/stack.py ==============

['SOBAH', 'SIMARA', 'ARONY', 'RIHAN', 'ISHAO']

ISHAO

['SOBAH', 'SIMARA', 'ARONY', 'RIHAN']

RIHAN

['SOBAH', 'SIMARA', 'ARONY']

2.

QUEUE IMPLEMENTATION USING LIST

queue = ["RISHAN", "SHIBHU", "BYHAN"]

queue.append("ROPHAN")

queue.append("ISHOB")

print(queue)

# Removes the first item

print(queue.pop(0))

print(queue)

# Removes the first item

print(queue.pop(0))

print(queue)

OUTPUT:

============== RESTART: C:/Users/abc/Desktop/pthon clg/queue.py ==============

['RISHAN', 'SHIBHU', 'BYHAN', 'ROPHAN', 'ISHOB']

RISHAN

['SHIBHU', 'BYHAN', 'ROPHAN', 'ISHOB']

SHIBHU

['BYHAN', 'ROPHAN', 'ISHOB']

**3.**

**Comparison of Running Time:**

For each function ***f(n)*** and time ***t***in the following table, determine the largest size ***n*** of a problem that can be solved in time ***t***, assuming that the algorithm to solve the problem takes ***f(n)*** microseconds.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***f(n)*** | 1 Second | 1 Minute | 1 Hour | 1 Day | 1 Month | 1 Year | 1 Century |
| log n | 210^6 | 26\*10^7 | 236\*10^8 | 2864\*10^8 | 22.5\*10^12 | 23.1\*10^13 | 23.1\*10^16 |
| √n | 1012 | 36\*1014 | 1296\*1014 | 7.4\*1021 | 6.7\*1024 | 9.9\*1026 | 9.9\*1032 |
| n | 106 | 6\*107 | 3.6 \*109 | 864\*108 | 2592\*109 | 31536\*109 | 3.1\*1015 |
| n log n | 62746 | 2.8\*106 | 1.3\*108 | 2.7\*109 | 7.1\*1010 | 7.9\*1011 | 6.8\*1013 |
| N^2 | 1000 | 7745 | 60000 | 293938 | 1609968 | 5615692 | 56175382 |
| N^3 | 100 | 391 | 1532 | 4420 | 13736 | 31593 | 146677 |
| 2^n | 19 | 25 | 31 | 36 | 41 | 44 | 51 |
| N! | 9 | 11 | 12 | 13 | 15 | 16 | 17 |

4. SOME DAA GENERAL PROBLEMS

Problem 01: Given n Boolean variables x1, x2, x3, .... xn we wish to print all possible combinations of truth values they can assume. For instance, if n=2, there are 4 possibilities: true, true; true, false; false, true and false, false. Write a program in python to accomplish this.

SOLUTION1: from itertools import product

n=int(input("enter value of n:  "))

print(list(product([True, False], repeat=n)))

OUTPUT1:================= RESTART: C:\Users\abc\Desktop\daa\exp3(1).py =================

enter value of n:  3

[(True, True, True), (True, True, False), (True, False, True), (True, False, False), (False, True, True), (False, True, False), (False, False, True), (False, False, False)]

>>>

Problem 02: Devise and implement an algorithm that inputs three integers and outputs them in nondecreasing order.

SOLUTION2:x=int(input("enter first no.:  "))

y=int(input("enter second no.:  "))

z=int(input("enter third no.:  "))

a=min(x,y,z)

c=max(x,y,z)

b=(x+y+z)-a-c

print("numbers of sorted",a,b,c)

OUTPUT2: enter first no.:  3

enter second no.:  45

enter third no.:  23

numbers of sorted 3 23 45

Problem 03: Devise and implement an algorithm that searches an unsorted array a[1:n] for the element x. If x occurs, then return a position in the array; else return zero.

SOLUTION 3: def search(arr, n, x):

    for i in range(0,n):

        if(arr[i]==x):

            return i;

        return -1;

arr=[10,20,30,40,50,60];

x=10;

n=len(arr);

result=search(arr, n, x)

if(result==-1):

    print("Element is not present in the array")

else:

    print("Element is present in the array at index ",result);

OUTPUT3: Element is present in the array at index  0

Problem 04: The factorial n! Has value 1 when n<=1 and value n\*(n-1)! When n>1. Write both recursive and an iterative algorithm to compute n!.

SOLUTION4: Iterative-

def factorial(n):

    if n < 0:

        return 0

    elif n == 0 or n == 1:

        return 1

    else:

        fact = 1

        while(n > 1):

            fact \*= n

            n -= 1

        return fact

num = 7;

print("Factorial of",num,"is",

factorial(num))

Recursive-

def factorial(n):

 return 1 if (n==1 or n==0) else n \* factorial(n - 1);

num = 7;

print("Factorial of",num,"is",

factorial(num))

OUTPUT 4:

Iterative-

Factorial of 7 is 5040

Recursive-

Factorial of 7 is 5040

Problem 05: The Fibonacci numbers are defined as f0= 0, f1= 1 and fi= fi-1 + fi-2 for i>1.

(a) List the values of f2 through f7.

(b) Write both a recursive and an iterative algorithm to compute fi .

SOLUTION5:  PART A

a=8   
first=0                                           
sec=1                                           
for x in range(2,a):   
    next=first+sec                             
    print(next,end=" ")   
    first=sec   
    sec=next

PART B

Number = int(input("\nPlease Enter the Number: "))

i = 0   
First\_Value = 0   
Second\_Value = 1   
              
while(i < Number):   
           if(i <= 1):   
                      Next = i   
           else:   
                      Next = First\_Value + Second\_Value   
                      First\_Value = Second\_Value   
                      Second\_Value = Next   
           print(Next)   
           i = i + 1

OUTPUT 5: Part A

1 2 3 5 8 13

Part B

Please Enter the Number: 5

0

1

1

2

3

 Problem 06: Ackermann's function A(m,n) is defined as follows:

A(m,n) =  n+1                   if m=0

A(m-1,1)                            if n=0

A(m-1,A(m,n-1))               otherwise

This function is studied because it grows very fast for small values of m and n. Write a recursive algorithm for computing this function. Then write a non recursive algorithm for computing it.

SOLUTION6: def A(m, n, s ="% s"):     
    #print(s % ("A(% d, % d)" % (m, n)))     
    if m == 0:     
        return n + 1    
    if n == 0:     
        return A(m - 1, 1, s)     
    n2 = A(m, n - 1, s % ("A(% d, %% s)" % (m - 1)))     
    return A(m - 1, n2, s)

print(A(1, 2))

OUTPUT6:

4

Problem 07: Implement an algorithm to solve the following problem: Given n, a positive integer, determine whether n is the sum of all its divisors, that is, whether n is the sum of all t such that 1<=t<n, and t divides n.

SOLUTION 7:  prompt = int(input("Enter an interger: "))

print("The divisors of the integer you entered are: ")    
divisor\_sum = 0 #updated line    
for i in range(1, prompt):    
    if(prompt%i==0):    
        print(i)    
        divisor\_sum+=i #calculate sum of all divisors

print("The sum of divisors " + str(divisor\_sum)) #print the sum of divisors

if n==divisor\_sum:    
    print("Valid ")    
else:    
    print("Not Valid")

OUTPUT 7:  Enter an interger: 6

The divisors of the integer you entered are:

1

2

3

The sum of divisors 6

Problem 08: If S is a set of n elements, the powerset of S is the set of all possible subsets of S. For example, if S = (a,b,c), then powerset (S) = {(), (a), (b), (c), (a,b), (b,c), (a,c), (a,b,c)}. Write a recursive algorithm to compute powerset(S).

SOLUTION 8: import math;

def printPowerSet(set,set\_size):

    pow\_set\_size = (int) (math.pow(2, set\_size));

    counter = 0;

    j = 0;

    for counter in range(0, pow\_set\_size):

        for j in range(0, set\_size):

            if((counter & (1 << j)) > 0):

                print(set[j], end = "");

        print("");

set = ['a', 'b', 'c'];

printPowerSet(set, 3);

OUTPUT 8:

a

b

ab

c

ac

bc

abc