**DESIGN AND ANALYSIS OF ALGORITHM -- CSA0679**

1.Fibonacci series using recursion:

#include <stdio.h>

int fibonacci(int n)

{

if (n <= 1)

{

return n;

}

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main()

{

int n, i;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

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

{

printf("%d ", fibonacci(i));

}

}

**Output:**

**Enter the number of terms: 5**

**Fibonacci Series: 0 1 1 2 3**

2.Write a program to check the given no is Armstrong or not.

#include <stdio.h>

int main() {

int number, originalNumber, remainder, result = 0;

printf("Enter an integer: ");

scanf("%d", &number);

originalNumber = number;

while (originalNumber != 0) {

remainder = originalNumber % 10;

result += remainder \* remainder \* remainder;

originalNumber /= 10;

}

if (result == number)

printf("%d is an Armstrong number.\n", number);

else

printf("%d is not an Armstrong number.\n", number);

return 0;

}

**Output**:

Enter an integer: 153

153 is an Armstrong number.

3. Write a program to find the GCD of two numbers

#include <stdio.h>

int main() {

int num1, num2, i, gcd;

printf("Enter two numbers: ");

scanf("%d %d", &num1, &num2);

for (i = 1; i <= num1 && i <= num2; ++i) {

if (num1 % i == 0 && num2 % i == 0) {

gcd = i;

}

}

printf("GCD of %d and %d is %d", num1, num2, gcd);

return 0;

}

**Output:**

Enter two numbers: 4 5

GCD of 4 and 5 is 1

4. Write a program to get the largest element of an array.

#include <stdio.h>

int main() {

int arr[] = {10, 20, 4, 45, 99};

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

int max = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] > max) {

max = arr[i];

}

}

printf("Largest element in the array is: %d", max);

return 0;

}

**Output**:

Largest element in the array is: 99

5. Write a program to find the Factorial of a number.

#include <stdio.h>

int main() {

int num,i,fact=1;

printf("Enter a positive integer: ");

scanf("%d", &num);

for (i = 1; i <= num; ++i) {

fact =fact\* i;

}

printf("Factorial of %d = %llu", num, fact);

return 0;

}

**Output**:

Enter a positive integer: 5

Factorial of 5 = 120

6. Write a program to check a number is a prime number or not.

#include <stdio.h>

int main() {

int num, i, flag = 0;

printf("Enter a number: ");

scanf("%d", &num);

for (i =1; i <= num; ++i) {

if (num % i == 0) {

flag=flag+1;

}

}

if (num == 1)

{

printf("1 is neither prime nor composite.");

}

else {

if (flag ==2 )

printf("%d is a prime number.", num);

else

printf("%d is not a prime number.", num);

}

return 0;

}

**Output**:

Enter a number: 7

7 is a prime number.

7. Write a program to perform Selection sort.

#include <stdio.h>

int main() {

int n, i, j, temp;

int arr[]={9,8,5,2,0,1};

n=sizeof(arr)/sizeof(arr[0]);

for (i = 0; i < n - 1; i++) {

for (j = i + 1; j < n; j++) {

if (arr[i] > arr[j]) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

printf("Sorted array in ascending order:\n");

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

printf(",%d",arr[i]);

}

return 0;

}

**Output**:

Sorted array in ascending order:

,0,1,2,5,8,9

8. Write a program to perform Bubble sort

#include <stdio.h>

int main() {

int n, i, j, temp;

int arr[] = {64, 34, 25, 12, 22, 11, 90};

n = sizeof(arr) / sizeof(arr[0]);

for (i = 0; i < n-1; i++) {

for (j = 0; j < n-i-1; j++) {

if (arr[j] > arr[j+1]) {

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

printf("Sorted array: \n");

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

printf("%d ", arr[i]);

}

return 0;

}

**Output**:

Sorted array:

11 12 22 25 34 64 90

9. Write a program for to multiply two Matrix

#include <stdio.h>

int main() {

int matrix1[2][2] = {{1, 2}, {3, 4}};

int matrix2[2][2] = {{1, 2}, {3, 4}};

int result[2][2] = {0};

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

for (int k = 0; k < 2; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

printf("Resultant Matrix:\n");

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

printf("%d ", result[i][j]);

}

printf("\n");

}

return 0;

}

**Output**:

Resultant Matrix:

7 10

15 22

10. Write a program for to check whether a given String is Palindrome or not.

#include <stdio.h>

#include <string.h>

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

int len = strlen(str);

int isPalindrome = 1;

for (int i = 0; i < len / 2; i++) {

if (str[i] != str[len - i - 1]) {

isPalindrome = 0;

break;

}

}

if (isPalindrome)

printf("%s is a palindrome.\n", str);

else

printf("%s is not a palindrome.\n");

return 0;

}

**Output:**

Enter a string: radar

radar is a palindrome.

11.Write a program for to copy one string to another

#include <stdio.h>

int main() {

char sour[] = "Hello, World!";

char des[50];

int i = 0;

while (sour[i] != '\0') {

des[i] = sour[i];

i++;

}

des[i] = '\0';

printf("Source string: %s\n", sour);

printf("Destination string: %s\n", des);

return 0;

}

**Output**:

Source string: Hello, World!

Destination string: Hello, World!

12. Write a Program to perform binary search

#include <stdio.h>

int main() {

int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}, n = 10, target = 23;

int first = 0, last = n - 1, middle;

while (first <= last) {

middle = (first + last) / 2;

if (arr[middle] < target)

first = middle + 1;

else if (arr[middle] == target) {

printf("%d found at position %d.\n", target, middle + 1);

return 0;

} else

last = middle - 1;

}

printf("Element not found in the array.\n");

return 0;

}

**Output**:

23 found at position 6.

13. Write a program to print the reverse of a string

{

char str[100];

int length, i;

printf("Enter a string: ");

scanf("%s", str);

length = strlen(str);

printf("Reverse of the string: ");

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

printf("%c", str[i]);

}

return 0;

}

**Output**:

Enter a string: saveetha

Reverse of the string: ahteevas

14. Write a program to find the length of a string

#include <stdio.h>

int main() {

char str[100];

int length = 0;

printf("Enter a string: ");

scanf("%s", str);

while (str[length] != '\0') {

length++;

}

printf("Length of the string: %d\n", length);

return 0;

}

**Output**:

Enter a string: saveetha

Length of the string: 8

15. Write a program to perform Strassen’s Matrix Multiplication.

#include <stdio.h>

int main() {

int a[2][2] = {{1, 2}, {3, 4}};

int b[2][2] = {{5, 6}, {7, 8}};

int c[2][2];

c[0][0] = a[0][0] \* b[0][0] + a[0][1] \* b[1][0];

c[0][1] = a[0][0] \* b[0][1] + a[0][1] \* b[1][1];

c[1][0] = a[1][0] \* b[0][0] + a[1][1] \* b[1][0];

c[1][1] = a[1][0] \* b[0][1] + a[1][1] \* b[1][1];

printf("Resultant Matrix:\n");

printf("%d %d\n", c[0][0], c[0][1]);

printf("%d %d\n", c[1][0], c[1][1]);

return 0;

}

**Output:**

Resultant Matrix:

19 22

43 50

16. Write a program to perform Merge Sort.

#include <stdio.h>

void merge(int arr[], int l, int m, int r) {

int i, j, k;

int n1 = m - l + 1, n2 = r - m;

int L[n1], R[n2];

for (i = 0; i < n1; i++)

L[i] = arr[l + i];

for (j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

i = 0;

j = 0;

k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int l, int r) {

if (l < r) {

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int arr\_size = sizeof(arr) / sizeof(arr[0]);

printf("Given array is \n");

for (int i = 0; i < arr\_size; i++)

printf("%d ", arr[i]);

printf("\n");

mergeSort(arr, 0, arr\_size - 1);

printf("\nSorted array is \n");

for (int i = 0; i < arr\_size; i++)

printf("%d ", arr[i]);

printf("\n");

}

**Output**:

Given array is 12 11 13 5 6 7

Sorted array is 5 6 7 11 12 13

17. Using Divide and Conquer strategy to find Max and Min value in the list

#include <stdio.h>

struct Pair {

int min;

int max;

};

struct Pair getMinMax(int arr[], int low, int high) {

struct Pair minmax, mml, mmr;

int mid;

if (low == high) {

minmax.min = arr[low];

minmax.max = arr[low];

return minmax;

}

if (high == low + 1) {

if (arr[low] > arr[high]) {

minmax.max = arr[low];

minmax.min = arr[high];

} else {

minmax.max = arr[high];

minmax.min = arr[low];

}

return minmax;

}

mid = (low + high) / 2;

mml = getMinMax(arr, low, mid);

mmr = getMinMax(arr, mid + 1, high);

if (mml.min < mmr.min) {

minmax.min = mml.min;

} else {

minmax.min = mmr.min;

}

if (mml.max > mmr.max) {

minmax.max = mml.max;

} else {

minmax.max = mmr.max;

}

return minmax;

}

int main() {

int arr[] = {1000, 11, 445, 1, 330, 3000};

int arr\_size = 6;

struct Pair minmax = getMinMax(arr, 0, arr\_size - 1);

printf("Minimum element is %d\n", minmax.min);

printf("Maximum element is %d\n", minmax.max);

return 0;

}

Output:

Minimum element is 1

Maximum element is 3000

18. Write a program to generate all the prime numbers.

#include <stdio.h>

int main() {

int n, i, j, isPrime;

printf("Enter a positive integer: ");

scanf("%d", &n);

printf("Prime numbers between 1 and %d are: ", n);

for (i = 2; i <= n; i++) {

isPrime = 1;

for (j = 2; j <= i / 2; j++) {

if (i % j == 0) {

isPrime = 0;

break;

}

}

if (isPrime == 1) {

printf("%d ", i);

}

}

return 0;

}

**Output**:

Enter a positive integer: 5

Prime numbers between 1 and 5 are: 2 3 5

19. Write a program to perform Knapsack problem using greedy techniques

#include <stdio.h>

void ks(int n, float weig[], float prof[], float cap);

int main() {

int n, i, j;

float weig[20], prof[20], ratio[20], temp, cap, total\_profit = 0.0;

printf("Enter the number of items: ");

scanf("%d", &n);

printf("Enter the weights and profits of each item:\n");

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

scanf("%f %f", &weig[i], &prof[i]);

ratio[i] = prof[i] / weig[i];

}

printf("Enter the capacity of the knapsack: ");

scanf("%f", &cap);

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

for (j = i + 1; j < n; j++) {

if (ratio[i] < ratio[j]) {

temp = ratio[j];

ratio[j] = ratio[i];

ratio[i] = temp;

temp = weig[j];

weig[j] = weig[i];

weig[i] = temp;

temp = prof[j];

prof[j] = prof[i];

prof[i] = temp;

}

}

}

ks(n, weig, prof, cap);

return 0;

}

void ks(int n, float weig[], float prof[], float cap) {

float x[20], tp = 0.0;

int i, j, u;

u = cap;

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

x[i] = 0.0;

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

if (weig[i] > u)

break;

else {

x[i] = 1.0;

tp += prof[i];

u -= weig[i];

}

}

if (i < n)

x[i] = u / weig[i];

tp += x[i] \* prof[i];

printf("The result vector is:\n");

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

printf("%f ", x[i]);

printf("\nThe maximum profit is: %f\n", tp);

}

**Output**:

Enter the number of items: 3

Enter the weights and profits of each item:

5 50

6 60

7 70

Enter the capacity of the knapsack: 5

The result vector is:

1.000000 0.000000 0.000000

The maximum profit is: 50.000000

20. Write a program to perform MST using greedy techniques

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <limits.h> // Include the limits.h header file

#define V 5

int minKey(int key[], bool mstSet[]) {

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++) {

if (!mstSet[v] && key[v] < min) {

min = key[v];

min\_index = v;

}

}

return min\_index;

}

void printMST(int parent[], int graph[V][V]) {

printf("Edge \tWeight\n");

for (int i = 1; i < V; i++)

printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);

}

void primMST(int graph[V][V]) {

int parent[V];

int key[V];

bool mstSet[V];

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

key[i] = INT\_MAX;

mstSet[i] = false;

}

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < V - 1; count++) {

int u = minKey(key, mstSet);

mstSet[u] = true;

for (int v = 0; v < V; v++) {

if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {

parent[v] = u;

key[v] = graph[u][v];

}

}

}

printMST(parent, graph);

}

int main() {

int graph[V][V] = {

{0, 2, 0, 6, 0},

{2, 0, 3, 8, 5},

{0, 3, 0, 0, 7},

{6, 8, 0, 0, 9},

{0, 5, 7, 9, 0}

};

primMST(graph);

return 0;

}

**Output**:

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5

21. Using Dynamic programming concept to find out Optimal binary search tree.

#include <stdio.h>

#include <limits.h>

int sum(int freq[], int i, int j) {

int s = 0;

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

s += freq[k];

}

return s;

}

int optimalBST(int keys[], int freq[], int n) {

int cost[n][n];

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

cost[i][i] = freq[i];

}

for (int L = 2; L <= n; L++) {

for (int i = 0; i <= n - L; i++) {

int j = i + L - 1;

cost[i][j] = INT\_MAX;

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

int c = ((r > i) ? cost[i][r - 1] : 0) +

((r < j) ? cost[r + 1][j] : 0) +

sum(freq, i, j);

if (c < cost[i][j]) {

cost[i][j] = c;

}

}

}

}

return cost[0][n - 1];

}

int main() {

int keys[] = {10, 12, 20};

int freq[] = {34, 8, 50};

int n = sizeof(keys) / sizeof(keys[0]);

printf("Cost of Optimal BST is %d\n", optimalBST(keys, freq, n));

return 0;

}

**Output**:

Cost of Optimal BST is 142

22. Using Dynamic programming techniques to find binomial coefficient of a given number

#include <stdio.h>

int min(int a, int b) {

return (a < b) ? a : b;

}

int binomialCoeff(int n, int k) {

int C[n + 1][k + 1];

int i, j;

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

for (j = 0; j <= min(i, k); j++) {

if (j == 0 || j == i)

C[i][j] = 1;

else

C[i][j] = C[i - 1][j - 1] + C[i - 1][j];

}

}

return C[n][k];

}

int main() {

int n = 5, k = 2;

printf("The binomial coefficient C(%d, %d) is %d\n", n, k, binomialCoeff(n, k));

return 0;

}

**Output**:

The binomial coefficient C(5, 2) is 10

23. Write a program to find the reverse of a given number

#include <stdio.h>

int main() {

int num, reversedNum = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &num);

while (num != 0) {

remainder = num % 10;

reversedNum = reversedNum \* 10 + remainder;

num /= 10;

}

printf("Reversed number = %d", reversedNum);

return 0;

}

**Output**:

Enter an integer: 45613

Reversed number = 31654

24. Write a program to find the perfect number.

#include <stdio.h>

int main() {

int num, sum = 0;

printf("Enter a number: ");

scanf("%d", &num);

for (int i = 1; i < num; i++) {

if (num % i == 0) {

sum += i;

}

}

if (sum == num) {

printf("%d is a perfect number.\n", num);

} else {

printf("%d is not a perfect number.\n", num);

}

return 0;

}

**Output**:

Enter a number: 6

6 is a perfect number.

25. Write a program to perform travelling salesman problem using dynamic programming

#include <stdio.h>

#define V 4

int graph[V][V] = {

{0, 10, 15, 20},

{10, 0, 35, 25},

{15, 35, 0, 30},

{20, 25, 30, 0}

};

int VISITED\_ALL = (1 << V) - 1;

int tsp(int mask, int pos) {

if (mask == VISITED\_ALL) {

return graph[pos][0];

}

int min\_cost = 99999;

for (int city = 0; city < V; city++) {

if ((mask & (1 << city)) == 0) {

int new\_cost = graph[pos][city] + tsp(mask | (1 << city), city);

if (new\_cost < min\_cost) {

min\_cost = new\_cost;

}

}

}

return min\_cost;

}

int main() {

int mask = 1;

int pos = 0;

printf("The minimum cost of the Travelling Salesman Problem is: %d\n", tsp(mask, pos));

return 0;

}

**Output**:

The minimum cost of the Travelling Salesman Problem is: 80

26. Write a program for the given pattern

If n=4 1

1 2

1 2 3

1 2 3 4

#include <stdio.h>

int main() {

int n = 4;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= i; j++) {

printf("%d ", j);

}

printf("\n");

}

return 0;

}

**Output**:

/tmp/OI6zNqlkEX.o

1

1 2

1 2 3

1 2 3 4

27. Write a program to perform Floyd’s algorithm

#include <stdio.h>

int main() {

int n, i, j, k;

printf("Enter the number of vertices: ");

scanf("%d", &n);

int graph[n][n];

printf("Enter the adjacency matrix:\n");

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

for (j = 0; j < n; j++)

scanf("%d", &graph[i][j]);

for (k = 0; k < n; k++)

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

for (j = 0; j < n; j++)

if (graph[i][k] + graph[k][j] < graph[i][j])

graph[i][j] = graph[i][k] + graph[k][j];

printf("Shortest distances between every pair of vertices:\n");

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

for (j = 0; j < n; j++)

printf("%d\t", graph[i][j]);

printf("\n");

}

return 0;

}

**Output**:

Enter the number of vertices: 22

Enter the adjacency matrix:

1 2 3 4

Shortest distances between every pair of vertices:

1 2

3 4

28. Write a program for pascal triangle

#include <stdio.h>

int main() {

int rows, coef = 1, space, i, j;

printf("Enter the number of rows: ");

scanf("%d", &rows);

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

for (space = 1; space <= rows - i; space++)

printf(" ");

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

if (j == 0 || i == 0)

coef = 1;

else

coef = coef \* (i - j + 1) / j;

printf("%4d", coef);

}

printf("\n");

}

return 0;

}

**Output**:

Enter the number of rows: 5

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

29. Write a program to find the optimal cost by using appropriate algorithm

#include <stdio.h>

int main() {

int cost[] = {10, 15, 20, 25, 30};

int n = sizeof(cost) / sizeof(cost[0]);

int min\_cost = cost[0];

for (int i = 1; i < n; i++) {

if (cost[i] < min\_cost) {

min\_cost = cost[i];

}

}

printf("The optimal cost is: %d", min\_cost);

return 0;

}

**Output**:

The optimal cost is: 10

30. Write a program to find the sum of digits.

#include <stdio.h>

int main() {

int num, digit, sum = 0;

printf("Enter a number: ");

scanf("%d", &num);

while (num != 0) {

digit = num % 10;

sum += digit;

num /= 10;

}

printf("Sum of digits: %d", sum);

return 0;

}

**Output**:

Enter a number: 153

Sum of digits: 9

31. Write a program to print a minimum and maximum value sequency for all the numbers in a list.

#include <stdio.h>

int main() {

int numbers[] = {4, 7, 2, 9, 1, 5};

int size = sizeof(numbers) / sizeof(numbers[0]);

int min = numbers[0];

int max = numbers[0];

for (int i = 1; i < size; i++) {

if (numbers[i] < min) {

min = numbers[i];

}

if (numbers[i] > max) {

max = numbers[i];

}

}

printf("Minimum value: %d\n", min);

printf("Maximum value: %d\n", max);

return 0;

}

**Output**:

Minimum value: 1

Maximum value: 9

32. Write a program to perform n Queens problem using backtracking.

#include <stdio.h>

#include <stdbool.h>

#define N 8

int board[N][N];

bool isSafe(int row, int col) {

int i, j;

for (i = 0; i < col; i++)

if (board[row][i])

return false;

for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j])

return false;

for (i = row, j = col; j >= 0 && i < N; i++, j--)

if (board[i][j])

return false;

return true;

}

bool solveNQueensUtil(int col) {

if (col >= N)

return true;

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

if (isSafe(i, col)) {

board[i][col] = 1;

if (solveNQueensUtil(col + 1))

return true;

board[i][col] = 0; // BACKTRACK

}

}

return false;

}

bool solveNQueens() {

if (solveNQueensUtil(0) == false) {

printf("Solution does not exist");

return false;

}

return true;

}

void printSolution() {

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

for (int j = 0; j < N; j++)

printf("%d ", board[i][j]);

printf("\n");

}

}

int main() {

if (solveNQueens()) {

printSolution();

}

return 0;

}

**Output**:

1 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

33. Write a program to inset a number in a list.

#include <stdio.h>

int main() {

int list[5] = {1, 2, 4, 5};

int number = 3;

int position = 2;

int i, n = 4;

for (i = n; i >= position; i--) {

list[i + 1] = list[i];

}

list[position] = number;

n++;

printf("List after inserting %d at position %d:\n", number, position);

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

printf("%d ", list[i]);

}

return 0;

}

**Output:**

**List after inserting 3 at position 2:**

**1 2 3 4 5**

34. Write a program to perform sum of subsets problem using backtracking

#include <stdio.h>

int total = 0;

int set[] = {10, 7, 5, 18, 12, 20, 15};

int n = 7;

int target = 35;

int include[7] = {0};

void sumOfSubsets(int i, int sum, int total) {

if (sum == target) {

printf("Subset found: ");

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

if (include[j] == 1) {

printf("%d ", set[j]);

}

}

printf("\n");

} else if (sum + set[i] <= target) {

include[i] = 1;

sumOfSubsets(i + 1, sum + set[i], total - set[i]);

}

if ((sum + total - set[i] >= target) && (sum + set[i + 1] <= target)) {

include[i] = 0;

sumOfSubsets(i + 1, sum, total - set[i]);

}

}

int main() {

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

total += set[i];

}

sumOfSubsets(0, 0, total);

return 0;

}

**Output**:

Subset found: 10 7 18 12

Subset found: 10 5 20

Subset found: 5 18 12 20

Subset found: 20 15

35. Write a program to perform graph coloring problem using backtracking.

#include <stdio.h>

int main() {

int graph[4][4] = {

{0, 1, 1, 1},

{1, 0, 1, 0},

{1, 1, 0, 1},

{1, 0, 1, 0}

};

int colors[4] = {0};

int m = 3; // Number of colors

for (int i = 0; i < 4; i++) {

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

if (graph[i][j] && colors[j] == colors[i]) {

colors[i] = (colors[i] + 1) % m;

j = -1;

}

}

}

printf("Colors assigned to vertices: \n");

for (int i = 0; i < 4; i++) {

printf("Vertex %d: Color %d\n", i, colors[i]);

}

return 0;

}

**Output**:

Colors assigned to vertices:

Vertex 0: Color 0

Vertex 1: Color 1

Vertex 2: Color 2

Vertex 3: Color 1

36. Write a program to compute container loader Problem.

#include <stdio.h>

int main() {

int container\_capacity = 1000;

int box1 = 300;

int box2 = 200;

int box3 = 400;

int total\_boxes\_loaded = 0;

int total\_weight\_loaded = 0;

while (total\_weight\_loaded + box1 <= container\_capacity) {

total\_weight\_loaded += box1;

total\_boxes\_loaded++;

}

while (total\_weight\_loaded + box2 <= container\_capacity) {

total\_weight\_loaded += box2;

total\_boxes\_loaded++;

}

while (total\_weight\_loaded + box3 <= container\_capacity) {

total\_weight\_loaded += box3;

total\_boxes\_loaded++;

}

printf("Total boxes loaded: %d\n", total\_boxes\_loaded);

printf("Total weight loaded: %d\n", total\_weight\_loaded);

return 0;

}

**Output:**

Total boxes loaded: 3

Total weight loaded: 900

37. Write a program to generate the list of all factor for n value.

#include <stdio.h>

int main() {

int n, i;

printf("Enter a positive integer: ");

scanf("%d", &n);

printf("Factors of %d are: ", n);

for (i = 1; i <= n; ++i) {

if (n % i == 0) {

printf("%d, ", i);

}

}

return 0;

}

**Output**:

Enter a positive integer: 5

Factors of 5 are: 1, 5,

38.Write a program to perform Assignment problem using branch and bound

#include <stdio.h>

#include <limits.h>

#define N 4

int costMatrix[N][N] = {{10, 2, 8, 5}, {3, 9, 7, 2}, {8, 3, 6, 2}, {2, 4, 8, 7}};

int rowMin[N], colMin[N], totalCost = 0;

void reduceMatrix(int reducedMatrix[N][N]) {

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

rowMin[i] = colMin[i] = INT\_MAX;

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

if (costMatrix[i][j] < rowMin[i]) rowMin[i] = costMatrix[i][j];

if (costMatrix[j][i] < colMin[i]) colMin[i] = costMatrix[j][i];

}

}

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

reducedMatrix[i][j] = costMatrix[i][j] - rowMin[i] - colMin[j];

}

void assignTask(int reducedMatrix[N][N], int row[N], int col[N], int rowSelected, int colSelected) {

row[rowSelected] = col[colSelected] = 1;

totalCost += costMatrix[rowSelected][colSelected];

for (int j = 0; j < N; j++)

if (!col[j]) assignTask(reducedMatrix, row, col, rowSelected + 1, j);

}

int main() {

int reducedMatrix[N][N], rowSelected[N] = {0}, colSelected[N] = {0};

reduceMatrix(reducedMatrix);

assignTask(reducedMatrix, rowSelected, colSelected, 0, 0);

printf("Total Cost: %d\n", totalCost);

return 0;

}

**Output**:

Total Cost: 32

39. Write a program for to perform liner search.

#include <stdio.h>

int main() {

int arr[] = {2, 4, 7, 1, 9, 5};

int target = 7;

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

int i, found = 0;

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

if (arr[i] == target) {

found = 1;

break;

}

}

if (found)

printf("Element found at index: %d", i);

else

printf("Element not found");

return 0;

}

**Output**:

Element found at index: 2

40.Write a program to find out Hamiltonian circuit Using backtracking method

#include <stdio.h>

#define V 5

int path[V];

void printSolution() {

for (int i = 0; i < V; i++) printf("%d ", path[i]);

printf("%d\n", path[0]);

}

int isSafe(int v, int graph[V][V], int path[], int pos) {

if (graph[path[pos - 1]][v] == 0) return 0;

for (int i = 0; i < pos; i++) if (path[i] == v) return 0;

return 1;

}

int hamCycleUtil(int graph[V][V], int path[], int pos) {

if (pos == V) return (graph[path[pos - 1]][path[0]] == 1) ? 1 : 0;

for (int v = 1; v < V; v++) {

if (isSafe(v, graph, path, pos)) {

path[pos] = v;

if (hamCycleUtil(graph, path, pos + 1) == 1) return 1;

path[pos] = -1;

}

}

return 0;

}

int hamCycle(int graph[V][V]) {

for (int i = 0; i < V; i++) path[i] = -1;

path[0] = 0;

if (hamCycleUtil(graph, path, 1) == 0) {

printf("No Hamiltonian Circuit exists\n");

return 0;

}

printSolution();

return 1;

}

int main() {

int graph[V][V] = {

{0, 1, 0, 1, 0},

{1, 0, 1, 1, 1},

{0, 1, 0, 0, 1},

{1, 1, 0, 0, 1},

{0, 1, 1, 1, 0}

};

hamCycle(graph);

return 0;

}

**Output**:

0 1 2 4 3 0