**Roll No…………….. Total No. of Pages:……**

**ST-4 (SET-VI)**

**6th SEMESTER 2023-24**

**CS192- Advanced Data Structures**

**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. In 1-Dimensional DP, what is the primary use of memoization?
   1. Reducing the time complexity of the problem
   2. Keeping track of the number of recursive calls
   3. **Storing the results of subproblems to avoid redundant computations**
   4. Converting recursive solutions to iterative solutions
2. Which of the following problems can be solved using grid DP?
   1. Finding the shortest path in a graph
   2. Calculating the Fibonacci sequence
   3. **Counting the number of ways to reach a target in a grid**
   4. Sorting an array
3. What is the space complexity of solving the Knapsack problem using dynamic programming?
   1. O(n)
   2. O(n log n)
   3. O(n^2)
   4. **O(nW), where W is the knapsack capacity**
4. Dynamic programming can be applied to problems with which type of time complexity?
   1. O(1)
   2. O(n)
   3. O(log n)
   4. **O(n^2)**
5. When solving multidimensional dynamic programming problems, what is the importance of identifying overlapping subproblems?
   1. **It helps reduce time complexity.**
   2. It ensures that the problem is solved bottom-up.
   3. It simplifies the problem statement.
   4. It indicates that the problem cannot be solved with dynamic programming.
6. In Dynamic Programming on Trees, what is the primary role of the "memo" table or data structure?
   1. **To store intermediate results and avoid redundant calculations**
   2. To represent the tree's structure
   3. To store the values of all tree nodes
   4. To keep track of the root node
7. Which greedy algorithm is used to find the shortest path in a weighted graph?
   1. **Dijkstra's algorithm**
   2. Prim's algorithm
   3. Kruskal's algorithm
   4. Bellman-Ford algorithm
8. What is the result of 12 | 9 in binary?
   1. **13**
   2. 8
   3. 12
   4. 9
9. What is the primary use of Fermat's little theorem?
   1. Finding prime numbers
   2. Solving Diophantine equations
   3. **Calculating modular inverses**
   4. Calculating Euler's totient function
10. Which C++ library is commonly used for handling big integers?
    1. <math.h>
    2. <cstdlib>
    3. <iomanip>
    4. **<boost/multiprecision/cpp\_int.hpp>**

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

11) What is the main goal of the Word Break Problem?

a) To break a given string into individual characters.

**b) To determine if a string can be segmented into a sequence of words from a dictionary.**

c) To find the longest subsequence of a string.

d) To reverse a given string.

12) Which of the following problems is closely related to the Subsets Sum problem and can also be solved using dynamic programming?

a) Longest Common Subsequence Problem

b) Longest Palindromic Subsequence Problem

**c) Coin Change Problem**

d) Optimal Binary Search Tree Problem

13) Consider a job scheduling problem with 4 jobs J1, J2, J3, J4 and with corresponding deadlines: ( d1, d2, d3, d4) = (4, 2, 4, 2). Which of the following is not a feasible schedule without violating any job schedule?

a) J2, J4, J1, J3

**b) J4, J1, J2, J3**

c) J4, J2, J1, J3

d) J4, J2, J3, J1

14) What does the following C expression do? x = (x<<1) + x + (x>>1);

a) Multiplies an integer with 7

**b) Multiplies an integer with 3.5**

c) Multiplies an integer with 3

d) Multiplies an integer with 8

15) What is the output of the following C++ code?

#include <iostream>

using namespace std;

int main() {

int n = 5;

int dp[6] = {0};

dp[0] = 1;

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

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

dp[j] += dp[j - i];

}

}

cout << "Number of ways to make change: " << dp[n] << endl;

return 0;

}

a) Number of ways to make change: 6

**b) Number of ways to make change: 7**

c) Number of ways to make change: 8

d) Number of ways to make change: 10

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) Evaluate the value of an arithmetic expression in Reverse Polish Notation (postfix notation).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | ["2", "1", "+", "3", "\*"] | ["6", "3", "2", "4", "+","-","\*"] | ["6", "3","/"] |
| **Output** | 9 | -18 | 2 |

Solution :

**#include <iostream>**

**#include <stack>**

**#include <vector>**

**using namespace std;**

**// Function to evaluate Reverse Polish Notation (RPN) expression**

**int evalRPN(vector<string>& tokens) {**

**stack<int> st; // Initialize a stack to hold operands**

**// Iterate through each token in the expression**

**for (string token : tokens) {**

**if (token == "+" || token == "-" || token == "\*" || token == "/") {**

**// If the token is an operator, pop the top two elements from the stack**

**int num2 = st.top(); st.pop();**

**int num1 = st.top(); st.pop();**

**// Perform the operation based on the operator**

**if (token == "+") st.push(num1 + num2);**

**else if (token == "-") st.push(num1 - num2);**

**else if (token == "\*") st.push(num1 \* num2);**

**else if (token == "/") st.push(num1 / num2);**

**} else {**

**// If the token is an operand, convert it to an integer and push it onto the stack**

**st.push(stoi(token));**

**}**

**}**

**// The final result will be left on the top of the stack**

**return st.top();**

**}**

**int main() {**

**vector<string> tokens = {"6", "3", "2", "4", "+","-","\*"};**

**cout << "Result: " << evalRPN(tokens) << endl; // Evaluate the RPN expression and display the result**

**return 0;**

**}**

Q17) Write a CPP program to print the top view of binary tree. Top view of a binary tree is the set of nodes visible when the tree is viewed from the top.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 10  / \  20 30  / \ / \  40 60 90 100 | 1  / \  2 3  \  4  \  5  \  6 | 10  / \  20 30 |
| **Output** | 40 20 10 30 100 | 2 1 3 6 | 20 10 30 |

Solution :

**#include <bits/stdc++.h>**

**using namespace std;**

**// Structure to represent a tree node**

**struct Node {**

**int data;**

**Node\* left;**

**Node\* right;**

**Node(int val) {**

**data = val;**

**left = right = NULL;**

**}**

**};**

**// Function to print the top view of a binary tree**

**void topView(Node\* root) {**

**if (!root)**

**return;**

**map<int, int> verticalMap; // Map to store vertical level and node data**

**queue<pair<Node\*, int>> q; // Queue for BFS traversal**

**q.push({root, 0});**

**while (!q.empty()) {**

**Node\* node = q.front().first;**

**int verticalLevel = q.front().second;**

**q.pop();**

**// Insert the node's data if not present in the map**

**if (verticalMap.find(verticalLevel) == verticalMap.end()) {**

**verticalMap[verticalLevel] = node->data;**

**}**

**// Push left child with decreased vertical level**

**if (node->left) {**

**q.push({node->left, verticalLevel - 1});**

**}**

**// Push right child with increased vertical level**

**if (node->right) {**

**q.push({node->right, verticalLevel + 1});**

**}**

**}**

**// Print the nodes in the top view**

**for (const auto& entry : verticalMap) {**

**cout << entry.second << " ";**

**}**

**}**

**int main() {**

**Node\* root = new Node(10);**

**root->left = new Node(20);**

**root->right = new Node(30);**

**root->left->right = new Node(60);**

**root->left->left = new Node(40);**

**root->right->left = new Node(90);**

**root->right->right = new Node(100);**

**cout << "Top view of the binary tree: ";**

**topView(root);**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) Write a C++ program to insert new element to MAX-Heap.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | heap={ 10, 5, 3, 2, 4 }  key=15 | heap={ 12,6,4,3,5 }  key=21 | heap={ 30,15,19 }  key=2 |
| **Output** | 15 5 10 2 4 3 | 21 6 12 3 5 4 | 30 15 19 2 |

Solution :

**#include <iostream>**

**using namespace std;**

**#define MAX 1000 // Max size of Heap**

**// Function to heapify ith node in a Heap**

**// of size n following a Bottom-up approach**

**void heapify(int arr[], int n, int i)**

**{**

**// Find parent**

**int parent = (i - 1) / 2;**

**if (arr[parent] > 0) {**

**// For Max-Heap**

**// If current node is greater than its parent**

**// Swap both of them and call heapify again**

**// for the parent**

**if (arr[i] > arr[parent]) {**

**swap(arr[i], arr[parent]);**

**// Recursively heapify the parent node**

**heapify(arr, n, parent);**

**}**

**}**

**}**

**// Function to insert a new node to the Heap**

**void insertNode(int arr[], int& n, int Key)**

**{**

**// Increase the size of Heap by 1**

**n = n + 1;**

**// Insert the element at end of Heap**

**arr[n - 1] = Key;**

**// Heapify the new node following a**

**// Bottom-up approach**

**heapify(arr, n, n - 1);**

**}**

**// A utility function to print array of size n**

**void printArray(int arr[], int n)**

**{**

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

**cout << arr[i] << " ";**

**cout << "\n";**

**}**

**int main()**

**{**

**// Array representation of Max-Heap**

**// 10**

**// / \**

**// 5 3**

**// / \**

**// 2 4**

**int arr[MAX] = { 10, 5, 3, 2, 4 };**

**int n = 5;**

**int key = 15;**

**insertNode(arr, n, key);**

**printArray(arr, n);**

**// Final Heap will be:**

**// 15**

**// / \**

**// 5 10**

**// / \ /**

**// 2 4 3**

**return 0;**

**}**