**CU / AIT / COMPUTER SCIENCE AND ENGINEERING / 4 / CSB-433**

****

**APEX INSTITUTE OF TECHNOLOGY**

**(AIT)**

Department of Computer Science and Engineering

B.E. – Computer Science and Engineering

SEMESTER : Sixth

SUBJECT NAME : Technical Training

SUBJECT CODE : 20-CSP-382

FACULTY : Dr. Prasenjit Das

**Contents of Course File**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Particular** | **Page No.** |
| 1 | University-Vision and Mission | 3 |
| 2 | Department-Vision and Mission | 4 |
| 3 | PEO | 5 |
| 4 | PO | 6 |
| 5 | PSO | 7 |
| 6 | Course Objectives | 7 |
| 7 | Course Outcomes | 7 |
| 8 | Mapping of COs/POs/PSOs | 7 -8 |
| 9 | Syllabus (As approved in BOS) ---(If Any Changes required, Approval Copy from DAA) | 9-10 |
| 10 | List of Experiments (Mapped with COs) | 11-13 |
| 11 | Experiment 1…10  Aim  Objective  Input/Apparatus Used  Procedure/Algorithm/Code  Observations/Outcome  Discussion  Question: Viva Voce | 14-66 |
| 12 | LAB MST  Question Paper (Mapped with COs)  Scheme of Evaluation  Sample Answer Sheets  CO-Wise Mark Sheets  List of Slow Learners and Advance Learners  Action Taken | 67-71 |
| 13 | Best Practices Adopted | 71 |
| 14 | Final Award Sheet | 72 |
| 15 | List of Slow Learners and Advance Learners | 73 |
| 16 | Overall Observation and Action Taken (Recommendation to BOS) |  |

Vision of Chandigarh University

To be globally recognized as a Centre of Excellence for Research, Innovation, Entrepreneurship and disseminating knowledge by providing inspirational learning to produce professional leaders for serving the society.

Mission of Chandigarh University

Providing world class infrastructure, renowned academicians and ideal environment for Research, Innovation, Consultancy and Entrepreneurship relevant to the society.

Offering programs & courses in consonance with National policies for nation building and meeting global challenges.

Designing Curriculum to match international standards, needs of Industry, civil society and for inculcation of traits of Creative Thinking and Critical Analysis as well as Human and Ethical values.

Ensuring students delight by meeting their aspirations through blended learning, corporate mentoring, professional grooming, flexible curriculum and healthy atmosphere based on co-curricular and extra-curricular activities.

Creating a scientific, transparent and objective examination/evaluation system to ensure an ideal certification.

Establishing strategic relationships with leading National and International corporates and universities for academic as well as research collaborations.

Contributing for creation of healthy, vibrant and sustainable society by involving in Institutional Social Responsibility (ISR) activities like rural development, welfare of senior citizens, women empowerment, community service, health and hygiene awareness and environmental protection

Vision of the Department

To be recognized as a centre of excellence for Computer Science & Engineering education and research, through effective teaching practices, hands-on training on cutting edge computing technologies and excellence in innovation, for creating globally aware competent professionals with strong work ethics whom would be proficient in implementing modern technology solutions and shall have entrepreneurial zeal to solve problems of organizations and society at large.

Mission of the Department

M1: To provide relevant, rigorous and contemporary curriculum and aligned assessment system to ensure effective learning outcomes for engineering technologies.

M2: To provide platform for industry engagement aimed at providing hands-on training on advanced technological and business skills to our students.

M3: To provide opportunities for collaborative, interdisciplinary and cutting-edge research aimed at developing solutions to real life problems

M4: To imbibe quest for innovation, continuous learning and zeal to pursue excellence through hard work and problem-solving approach

M5: To foster skills of leadership, management, communication, team spirit and strong professional ethics in all academic and societal endeavours of our students

Program Education Objectives

**PE01** To be able to explore areas of research, technology application & innovation and make a positive impact in different types of institutional settings such as corporate entities, government bodies, NGOs, inter-government organizations, & start-ups.

**PEO2** To be able to design, and implement technology and computing solutions to the organizational problems, effectively deploy knowledge of engineering principles, demonstrate critical thinking skills&make the intellectual connections between quantitative and qualitative tools, theories and context to solve the organizational problems

**PEO3** To be able to work with, lead & engage big and small teams comprising diverse people in terms of gender, nationality, region, language, culture & beliefs. To understand stated and unstated differences of views, beliefs & customs in diverse & inter disciplinary team settings

**PEO4** To be able to continuously learn and update one’s knowledge, engage in lifelong learning habits and acquire latest knowledge to perform in current work settings

**PEO5** To continuously strive for justice, ethics, equality, honesty, and integrity both in personal and professional pursuits. Able to understand and conduct in a way that is responsible and respectful.

Program Outcomes

**PO1 Engineering Knowledge**: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

**PO2 Problem Analysis**: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

**PO3 Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

**PO4 Conduct investigations of complex problems** using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

**PO5 Modern Tool Usage**: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6 The Engineer and Society**: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

**PO7 Environment and Sustainability**: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

**PO8 Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

**PO9 Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.

**PO10 Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

**PO11 Project Management and Finance**: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments

**PO12 Life-long Learning**: Recognize the need for and have the preparation and ability to Engage in independent and life- long learning in the broadest context of technological Change.

Program Specific (Learning) Outcome for Information Security

Specifically, the Program Specific (Learning) Outcomes (PSOs) of the Information Security program are that upon graduation students will be able to:

**PSO1** Analyze and evaluate systems with respect to maintaining operations in the presence of risks and threats, and also communicate the human role in security systems with an emphasis on ethics, social engineering vulnerabilities and training.

**PSO2** Conduct an Information Security risk assessment an Audit and troubleshoot Information Security systems using Cryptographic measures

**PSO3** Be able to assess the Ethical Ramification of working in Information Security, information assurance, and cyber/computer forensics software/tools.

**PSO4** Design and develop a security architecture and operational and strategic cyber security strategies and policies.

COURSE OBJECTIVES

# The Course aims to:

1. To provide hands-on for programming languages C and C++.
2. To provide in-depth knowledge of the various techniques of problem solving
3. To make the students industry ready and take up challenges of IT Industry

COURSE OUTCOMES

On completion of this course, the students shall be able to

1. Compare and contrast programming skills in languages such as C and C++.
2. Interpret the concepts of various data structures for real-time problems.
3. Sketch an efficient algorithm to solve the real-time problem.
4. Design a solution for existing coding challenges in the field of IT and improve employability.
5. Correlate and develop a solution to improve the complexity of the real-time problem.

Mapping of COs/POs/PSOs

|  |  |  |
| --- | --- | --- |
| **Mapping Between COs and Pos** | | |
| **S.No.** | **Course Outcome (CO)** | **Mapped Programme Outcome (PO)** |
| 1 | Compare and contrast programming skills in languages such as C and C++. | 1. to 5 |
| 2 | Interpret the concepts of various data structures for real-time problems. | 1. to 5 |
| 3 | Sketch an efficient algorithm to solve the real-time problem | 1 to 5 |
| 4. | Design a solution for existing coding challenges in the field of IT and improve employability | 1 to 5 |
| 5. | Correlate and develop a solution to improve the complexity of the real-time problem | 1to 5 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Engineering Knowledge | Problem analysis | Design/development of solutions | Conduct investigations of complex problems | Modern tool usage | The engineer and society | Environment and sustainability | Ethics | Individual or team work | Communication | Project management and finance | Life-long Learning |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| CSP-382 | CO1 | 2 | 2 | 3 | 3 | 2 |  |  |  |  | 1 |  | 2 |
| Technical Training | CO2 | 2 | 2 | 3 |  | 1 |  |  |  | 2 | 3 |  |  |
| CO3 | 3 | 2 | 3 | 3 | 3 | 2 | 1 |  | 3 |  | 2 | 3 |
| CO4 | 2 | 2 | 2 | 2 | 2 | 2 |  |  |  | 3 |  |  |
| CO5 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 |  |  | 1 | 2 |

1=addressed to small extent

2= addressed significantly

3=major part of course

Syllabus

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **20-CSP-382** | **INCIDENT RESPONSE and THREAT HUNTING LAB** | **L** | **T** | **P** | **S** | **C** | **CH** |
| Version 1.00 | | 0 | 0 | 2 |  |  |  |
| Pre-requisites/ Exposure | 1. Knowledge of Basics of Programming. 2. Knowledge of basic data structure. | | | | | | |

**COURSE DESCRIPTION**

The course provides hands on for technical coding. Experiments/programs are aimed to enhance the skills of programming in the languages of C and C++. Students who have already done basic coding will be exposed to problems generally asked in various placement activities. The idea is to make the students industry ready.

**TEXT BOOKS**

**T1** Reema Thareja, "Data Structures Using C", Oxford Publications, 2016.

**T2** Sartaj Sahni, "Fundamentals of Algorithms",University Press, 2017.

**REFERENCE BOOKS**

**R1** E Balaguruswamy , “Object Oriented Programming”, McGraw Hill 2020

**R2** Reema Thareja, "programming in C", Oxford Publications, 2016.

**COURSE CONTENT**

# Unit I 8 Contact Hours

1. To perform operations related to datastructures which are in non continous memmory blocks
2. Use a particular Data Structure to implement other data structures.
3. Solve problems related to character and string arrays in C++.

# Unit II 8 Contact Hours

1. To be able to write codes and perform operations on Non –linear Data Structures.
2. Apply basic problem solving approaches like divide and conquer and Dynamic Programming

# Unit III 8 Contact Hours

1. To solve the coding challenges as asked in various Placement activities
2. Relate give coding challenge to a basic problem solving approach

# MODE OF EVALUATION: The performance of students is evaluated as follows:

|  |  |  |
| --- | --- | --- |
|  | **Theory** | |
| **Components** | **Continuous Internal Assessment (CAE)** | **Semester End Examination (SEE)** |
| **Marks** | 60 | 40 |
| **Total Marks** | 100 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO vs PO/PSO** | | **PO1** | **PO2** | **PO3** | | **PO4** | **PO5** | **PO6** | | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | | **PO12** | **PSO1** | **PSO2** | | **PSO3** | **PSO4** | |
|  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
| **CO1** | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
|  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
| **CO2** | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
|  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
| **CO3** | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
|  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
| **CO4** | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
|  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
| **Target** | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
|  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  | |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | Lecture Plan -Practical | | | | | | | | | |  | | | |  | | |
| **Unit No** | **ExperimentNo** | | | | **Experiment Name** | | | | **Text/ Reference Books** | | | | | | **Pedagogical Tool\*\*** | | | | **Mapped with CO** | | |
|  |  | | | |  | | | |  | | | | | |  | | | |  | | |

Experiment 1

Experiment 1:A data structure needs to be implemented in such a way that we have the references i.e. the addresses of the values. None of the addresses are in continuous memory block. Each time a new value needs to be stored, we need to allocate memory. Write a program to implement the following:

* + - 1. Addition of a new value at a given position
      2. Print all the values in the list
      3. Delete a given value from a a location

CO Attained: CO2 and CO3

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtPos(int x, int pos){

int n = getSize();

if(pos>n || pos<0){return ;}

if(pos==0){insertAtBegin(x); return ;}

Node \*p1 = head;

while(--pos){

p1 = p1->next;

}

Node \*n1 = new Node(x);

n1->next = p1->next;

p1->next = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

void deleteBegin(){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

head = head->next;

}

void deletValue(int x){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

Node \*prev=NULL , \*curr=head;

bool flag=true;

while(curr){

if(curr->val == x){

flag=false;

Node \*temp = curr;

prev->next = curr->next;

curr->next = prev;

// delete(curr);

// delete(temp);

}

prev = curr;

curr = curr->next;

}

if(flag){

cout<<"There exist no element which is asked to delete.\n";

}else{

cout<<"Deleted\n";

}

}

void deleteAtPos(int x, int pos){

int n = getSize();

if(pos>n || pos<=0){return ;}

if(pos==1){deleteBegin(); return ;}

Node \*p1 = head, \*prev=nullptr;

while(--pos){

prev = p1;

p1 = p1->next;

}

if(p1->val==x){

prev->next = p1->next;

free(p1);

}

}

void deleteEnd(){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

Node \*temp = head;

while(temp->next->next){

temp = temp->next;

}

temp->next = NULL;

}

void sorting(){

Node \*temp1 = head, \*temp2 = head;

while(temp1->next){

temp2 = temp1->next;

while(temp2){

if(temp2->val < temp1->val){

int temp = temp2->val;

temp2->val = temp1->val;

temp1->val = temp;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

Node\* getHead(){return this->head;}

int getSize(){

if(!head)return 0;

if(!head->next)return 1;

Node \*curr = head;

int ans=0;

while(curr){

curr = curr->next;

ans++;

}

return ans;

}

};

int main(){

LL l1;

l1.insertAtEnd(11); //

l1.insertAtEnd(13); //

l1.insertAtEnd(12); //

l1.insertAtEnd(19); //

l1.insertAtEnd(12); //

l1.insertAtEnd(13);

l1.insertAtPos(999,1);

l1.deleteAtPos(12,2);

l1.printList();

return 0;

}

Experiment 2: Two lists are to be maintained containing integer values. The values are not stored in continuous memory locations. Let one list be 7->1->5 and another list be 5->9->2. Create a third list such that the third list contains the sum of the first two list i.e 517+295 =812 and save it as 2->1>8.

Node\* addTwoNumbers(Node\* l1, Node\* l2) {

if(!l1)return NULL;

if(!l2)return NULL;

Node\* ptr = new Node(0);

Node\* prev = ptr;

int cary = 0;

while(l1 || l2){

int sum = (l1 != NULL ? l1->val : 0) + (l2 != NULL ? l2->val : 0) + cary;

cary = sum/10;

Node\* temp = new Node(sum%10);

prev->next = temp;

prev = prev->next;

if(l1)l1=l1->next;

if(l2)l2=l2->next;

}

if(cary){

Node\* temp = new Node(cary);

prev->next = temp;

}

return ptr->next;

}

Experiment 3: Names of persons are saved in a list. Create function to check if the names are palindrome or not.

CO attained CO1, CO3, CO5

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int data;

Node(int d) { data = d; }

Node\* ptr;

};

// Function to check if the linked list

// is palindrome or not

bool isPalin(Node\* head)

{

// Temp pointer

Node\* slow = head;

// Declare a stack

stack<int> s;

// Push all elements of the list

// to the stack

while (slow != NULL) {

s.push(slow->data);

// Move ahead

slow = slow->ptr;

}

// Iterate in the list again and

// check by popping from the stack

while (head != NULL) {

// Get the top most element

int i = s.top();

// Pop the element

s.pop();

// Check if data is not

// same as popped element

if (head->data != i) {

return false;

}

// Move ahead

head = head->ptr;

}

return true;

}

// Driver Code

int main()

{

// Addition of linked list

Node one = Node(1);

Node two = Node(2);

Node three = Node(3);

Node four = Node(2);

Node five = Node(1);

// Initialize the next pointer

// of every current pointer

five.ptr = NULL;

one.ptr = &two;

two.ptr = &three;

three.ptr = &four;

four.ptr = &five;

Node\* temp = &one;

// Call function to check palindrome or not

int result = isPalin(&one);

if (result == 1)

cout << "isPalindrome is true\n";

else

cout << "isPalindrome is false\n";

return 0;

}

Experiment 4: You are in the process of creating a text editor (like notepad /Ms Word). The text editor saves particular word in form a string. We need to provide the feature of Find and Replace a particular substring in the given text editor.

CO Attained: CO2 and CO4

#include <stdio.h>

#include <math.h>

#include <stdbool.h>

int calc\_hash(char \*arr, int a, int n){

int pm = 2;

int ans = 0;

int k = n;

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

ans += (arr[i])\*(pow(pm,(k--)-1));

}

return ans;

}

int main()

{

char str[] = "CABDCAC";

char pat[] = "ABD";

int n = sizeof(str)/sizeof(str[0])-1;

int m = sizeof(pat)/sizeof(pat[0])-1;

int hash = calc\_hash(pat,0,m);

printf("Pattern Hash : %d\n",hash);

bool flag = false;

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

int curr\_hash = calc\_hash(str,i,m);

if(hash == curr\_hash){

flag = true;

break;

}

}

if(flag)printf("Found");

else printf("Not Found");

return 0;

}

Experiment 5: Characters are sorted in a form of a list (non continuous). We have to check if the list of characters has a given character in it or not.

CO Attained: CO3, CO4 and CO5

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

bool isFound(int x){

Node \*temp = head;

while(temp){

if(temp->val == x){

return true;

}

temp = temp->next;

}

return false;

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}x`

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

};

int main(){

LL l1,l2,l3;

l1.insertAtEnd(1);

l1.insertAtEnd(5);

l1.insertAtEnd(7);

cout<<(l1.isFound(12) ? "Found\n" : "Not Found\n");

return 0;

}

Experiment 6 : Three list are maintained with the integer roll nos of the students. One list contains the roll nos and name of the students, second list contains the roll nos and the course name (BE, BCA etc) for the student and the third list contains the roll no. and CGPA of the student. Find a way to check the common roll nos in the 3 lists. Note: the 3 lists are sorted by the roll nos.

Cos attained:

CO2, CO3 and CO4

#include<bits/stdc++.h>

using namespace std;

int isPresent(vector<int>&arr, int k){

int n = arr.size();

int l=0, r=n-1, ans=0;

while(l<=r){

int m = (l+r)/2;

if(arr[m]==k){

ans = 1;

break;

}else if(arr[m]<k){

l=m+1;

}else{

r=m-1;

}

}

return ans;

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main(){

vector<int>v1 = {1,2,3,4,5,8,9};

vector<int>v2 = {2,4,8,9,11,13};

vector<int>v3 = {1,2,3,5,9,12,18,18};

vector<int>ans;

int n1=v1.size(), n2=v2.size(), n3=v3.size();

int n = min({n1,n2,n3});

if(n1==n){

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

if(isPresent(v2,v1[i]) && isPresent(v3,v1[i])){

ans.push\_back(v1[i]);

}

}

}else if(n2==n){

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

if(isPresent(v1,v2[i]) && isPresent(v3,v2[i])){

ans.push\_back(v2[i]);

}

}

}else{

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

if(isPresent(v2,v3[i]) && isPresent(v1,v3[i])){

ans.push\_back(v3[i]);

}

}

}

print1d(ans);

return 0;

}

Experiment 7 : A set of strings are saved in a data structure. Represent the strings in form a non linear data structure in such a way that the searching takes the minimal time

CO attained: CO2, CO4

#include <bits/stdc++.h>

using namespace std;

class Node{

public:

Node \*alpha[26] = {0};

bool end = false;

bool containsKey(char ch){

return alpha[ch-'a'] != NULL;

}

void setNode(char ch, Node \*node){

alpha[ch-'a'] = node;

}

Node \*getNode(char ch){

return alpha[ch-'a'];

}

void setEnd(){

end = true;

}

bool isEnd(){

return end;

}

};

class Trie{

Node \*root;

public:

Trie(){

root = new Node();

}

void insertWord(string s){

Node \*n1 = root;

int n = s.size();

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

if( !(n1->containsKey(s[i])) ){

n1->setNode(s[i],new Node());

}

n1 = n1->getNode(s[i]);

}

n1->setEnd();

}

bool searchWord(string s){

int n = s.size();

Node \*n1 = root;

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

if( !(n1->containsKey(s[i]))){

return false;

}

n1 = n1->getNode(s[i]);

}

return n1->isEnd();

}

};

int main()

{

Trie t1;

t1.insertWord("Hello");

t1.insertWord("Hell");

t1.insertWord("Heed");

t1.insertWord("Heap");

t1.insertWord("Help");

t1.insertWord("Helo");

cout<<t1.searchWord("Hello")<<"\n";

cout<<t1.searchWord("Hell");

return 0;

}

Experiment 8: Without comparing the integer values in an array, sort the array of integers in ascending order.

CO ATTAINED: CO2, CO4

#include <bits/stdc++.h>

using namespace std;

void count\_sort(vector<int>&v){

map<int,int>um;

for(auto &x : v){

um[x]++;

}

int itr = 0;

for(auto &x : um){

int num = x.first;

int times = x.second;

while(times--){

v[itr++] = num;

}

}

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main()

{

vector<int>v = {2,5,3,4,1,6,7,3,5,9};

print1d(v);

count\_sort(v);

print1d(v);

return 0;

}

Experiment 9: List of Marks obtained by students of your class is maintained in form a linked List. Apply a sorting algorithm with complexity O (n^2) to sort the list.

CO Attained: CO3 and CO5

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

void sorting(){

Node \*temp1 = head, \*temp2 = head;

while(temp1->next){

temp2 = temp1->next;

while(temp2){

if(temp2->val < temp1->val){

int temp = temp2->val;

temp2->val = temp1->val;

temp1->val = temp;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

};

int main(){

LL l1;

l1.insertAtEnd(11); // 11

l1.insertAtEnd(12); // 11 12

l1.insertAtEnd(13); // 11 12 13

l1.insertAtEnd(14); // 11 12 13 14

l1.insertAtBegin(90); // 90 11 12 13 14 15

l1.insertAtBegin(80); // 80 90 11 12 13 14 15

l1.sorting();

l1.printList();cout<<"\n"; // 11->13->14

return 0;

}

Experiment 10 : Two arrays are maintained such that the marks are in descending order. We need to reverse the two arrays using another data structure so that the arrays are in ascending order. Hint: Use LIFO

CO Attained: CO1, CO3, CO4

#include<bits/stdc++.h>

using namespace std;

void reverse\_using\_stack(vector<int>&v){

stack<int>st;

for(auto &x : v){

st.push(x);

}

int itr=0;

while(st.size()){

v[itr++] = st.top();

st.pop();

}

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main(){

vector<int>v1 = {5,4,3,2,1};

vector<int>v2 = {9,8,7,6,5};

reverse\_using\_stack(v1);

reverse\_using\_stack(v2);

print1d(v1);

print1d(v2);

return 0;

}

Experiment 11: A thief enters a house. He is carrying a sack of some capacity say ‘m’. He can fill the sack with objects in the house and can carry a fraction of the object. The weights of each object are given by wi. For carrying each of the object in the sack, he earns a profit of pi. Write a code to maximize his profit.

CO Attained : CO2, CO4, CO5

#include <bits/stdc++.h>

using namespace std;

// Returns the value of maximum profit

int knapSackRec(int W, int wt[], int val[], int i, int\*\* dp)

{

// base condition

if (i < 0)

return 0;

if (dp[i][W] != -1)

return dp[i][W];

if (wt[i] > W) {

// Store the value of function call

// stack in table before return

dp[i][W] = knapSackRec(W, wt, val, i - 1, dp);

return dp[i][W];

}

else {

// Store value in a table before return

dp[i][W] = max(val[i]

+ knapSackRec(W - wt[i], wt, val,

i - 1, dp),

knapSackRec(W, wt, val, i - 1, dp));

// Return value of table after storing

return dp[i][W];

}

}

int knapSack(int W, int wt[], int val[], int n)

{

// double pointer to declare the

// table dynamically

int\*\* dp;

dp = new int\*[n];

// loop to create the table dynamically

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

dp[i] = new int[W + 1];

// loop to initially filled the

// table with -1

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

for (int j = 0; j < W + 1; j++)

dp[i][j] = -1;

return knapSackRec(W, wt, val, n - 1, dp);

}

// Driver Code

int main()

{

int val[] = { 60, 100, 120 };

int wt[] = { 10, 20, 30 };

int W = 50;

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

cout << knapSack(W, wt, val, n);

return 0;

}

Experiment 12: Write a code for the experiment above with the case that the thief cannot choose any object in its fraction. He can either pick the object or he can leave the object.

CO Attained: CO3, CO4

#include <stdio.h>

int n = 5; /\* The number of objects \*/

int c[10] = {12, 1, 2, 1, 4}; /\* c[i] is the \*COST\* of the ith object; i.e. what

YOU PAY to take the object \*/

int v[10] = {4, 2, 2, 1, 10}; /\* v[i] is the \*VALUE\* of the ith object; i.e.

what YOU GET for taking the object \*/

int W = 15; /\* The maximum weight you can take \*/

void simple\_fill() {

int cur\_w;

float tot\_v;

int i, maxi;

int used[10];

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

used[i] = 0; /\* I have not used the ith object yet \*/

cur\_w = W;

while (cur\_w > 0) { /\* while there's still room\*/

/\* Find the best object \*/

maxi = -1;

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

if ((used[i] == 0) &&

((maxi == -1) || ((float)v[i]/c[i] > (float)v[maxi]/c[maxi])))

maxi = i;

used[maxi] = 1; /\* mark the maxi-th object as used \*/

cur\_w -= c[maxi]; /\* with the object in the bag, I can carry less \*/

tot\_v += v[maxi];

if (cur\_w >= 0)

printf("Added object %d (%d$, %dKg) completely in the bag. Space left: %d.\n", maxi + 1, v[maxi], c[maxi], cur\_w);

else {

printf("Added %d%% (%d$, %dKg) of object %d in the bag.\n", (int)((1 + (float)cur\_w/c[maxi]) \* 100), v[maxi], c[maxi], maxi + 1);

tot\_v -= v[maxi];

tot\_v += (1 + (float)cur\_w/c[maxi]) \* v[maxi];

}

}

printf("Filled the bag with objects worth %.2f$.\n", tot\_v);

}

int main(int argc, char \*argv[]) {

simple\_fill();

return 0;

}

Experiment 13: Implement a Stack using Two Queues. Namely Push and PoP operations

CO Attained : CO2, CO3

#include <stdio.h>

#include <stdlib.h>

void push1(int);

void push2(int);

int pop1();

int pop2();

void enqueue();

void dequeue();

void display();

void create();

int st1[100], st2[100];

int top1 = -1, top2 = -1;

int count = 0;

void main()

{

int ch;

printf("\n1 - Enqueue element into queue");

printf("\n2 - Dequeu element from queue");

printf("\n3 - Display from queue");

printf("\n4 - Exit");

create();

while (1)

{

printf("\nEnter choice");

scanf("%d", &ch);

switch (ch)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("Wrong choice");

}

}

}

/\*Function to create a queue\*/

void create()

{

top1 = top2 = -1;

}

/\*Function to push the element on to the stack\*/

void push1(int data)

{

st1[++top1] = data;

}

/\*Function to pop the element from the stack\*/

int pop1()

{

return(st1[top1--]);

}

/\*Function to push an element on to stack\*/

void push2(int data)

{

st2[++top2] = data;

}

/\*Function to pop an element from th stack\*/

int pop2()

{

return(st2[top2--]);

}

/\*Function to add an element into the queue using stack\*/

void enqueue()

{

int data, i;

printf("Enter data into queue");

scanf("%d", &data);

push1(data);

count++;

}

/\*Function to delete an element from the queue using stack\*/

void dequeue()

{

int i;

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

{

push2(pop1());

}

pop2();

count--;

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

{

push1(pop2());

}

}

/\*Function to display the elements in the stack\*/

void display()

{

int i;

for (i = 0;i <= top1;i++)

{

printf(" %d ", st1[i]);

}

}

Experiment 14: Flood fill (also known as seed fill) is an algorithm that determines the area connected to a given node in a multi-dimensional array. It is used in the “bucket” fill tool of a paint program to fill connected, similarly colored areas with a different color and in games such as Go and Minesweeper for determining which pieces are cleared. When applied on an image to fill a particular bounded area with color, it is also known as boundary fill.

CO Attained: CO2, CO3, CO4

#include <iostream>

#include <queue>

#include <iomanip>

using namespace std;

// Below arrays detail all eight possible movements

int row[] = { -1, -1, -1, 0, 0, 1, 1, 1 };

int col[] = { -1, 0, 1, -1, 1, -1, 0, 1 };

// check if it is possible to go to pixel (x, y) from the

// current pixel. The function returns false if the pixel

// has a different color, or it's not a valid pixel

bool isSafe(vector<vector<char>> const &mat, int x, int y, char target)

{

return (x >= 0 && x < mat.size()) && (y >= 0 && y < mat[0].size())

&& mat[x][y] == target;

}

// Flood fill using BFS

void floodfill(vector<vector<char>> &mat, int x, int y, char replacement)

{

// base case

if (mat.size() == 0) {

return;

}

// create a queue and enqueue starting pixel

queue<pair<int, int>> q;

q.push({x, y});

// get the target color

char target = mat[x][y];

// target color is same as replacement

if (target == replacement) {

return;

}

// break when the queue becomes empty

while (!q.empty())

{

// dequeue front node and process it

pair<int, int> node = q.front();

q.pop();

// (x, y) represents the current pixel

int x = node.first, y = node.second;

// replace the current pixel color with that of replacement

mat[x][y] = replacement;

// process all eight adjacent pixels of the current pixel and

// enqueue each valid pixel

for (int k = 0; k < 8; k++)

{

// if the adjacent pixel at position (x + row[k], y + col[k]) is

// is valid and has the same color as the current pixel

if (isSafe(mat, x + row[k], y + col[k], target))

{

// enqueue adjacent pixel

q.push({x + row[k], y + col[k]});

}

}

}

}

// Utility function to print a matrix

void printMatrix(vector<vector<char>> const &mat)

{

for (int i = 0; i < mat.size(); i++)

{

for (int j = 0; j < mat[0].size(); j++) {

cout << setw(3) << mat[i][j];

}

cout << endl;

}

}

int main()

{

// matrix showing portion of the screen having different colors

vector<vector<char>> mat =

{

{ 'Y', 'Y', 'Y', 'G', 'G', 'G', 'G', 'G', 'G', 'G' },

{ 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'G', 'X', 'X', 'X' },

{ 'G', 'G', 'G', 'G', 'G', 'G', 'G', 'X', 'X', 'X' },

{ 'W', 'W', 'W', 'W', 'W', 'G', 'G', 'G', 'G', 'X' },

{ 'W', 'R', 'R', 'R', 'R', 'R', 'G', 'X', 'X', 'X' },

{ 'W', 'W', 'W', 'R', 'R', 'G', 'G', 'X', 'X', 'X' },

{ 'W', 'B', 'W', 'R', 'R', 'R', 'R', 'R', 'R', 'X' },

{ 'W', 'B', 'B', 'B', 'B', 'R', 'R', 'X', 'X', 'X' },

{ 'W', 'B', 'B', 'X', 'B', 'B', 'B', 'B', 'X', 'X' },

{ 'W', 'B', 'B', 'X', 'X', 'X', 'X', 'X', 'X', 'X' }

};

// start node

int x = 3, y = 9; // having target color `X`

// replacement color

char replacement = 'C';

// replace the target color with a replacement color

floodfill(mat, x, y, replacement);

// print the colors after replacement

printMatrix(mat);

return 0;

}

Experiment 15: The hierarchy of a family is to be maintained in such a way that the starting from Great grandfather, we need to depict till the grandsons/granddaughters. Note that at no level, the parent has more than 2 children. Depict such hierarchy using a suitable data structure. In such data structure, apply the following traversals and Count the number of Leaf and Non-leaf nodes:

1. In-order
2. Pre-order
3. Post-order
4. Level Order
5. Diagonal Order

CO Attained: CO1, CO2, CO3, CO4, CO5

In-order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node\* left;

struct node\* right;

};

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Given a binary tree, print its nodes in inorder\*/

void printInorder(struct node\* node)

{

if (node == NULL)

return;

/\* first recur on left child \*/

printInorder(node->left);

/\* then print the data of node \*/

printf("%d ", node->data);

/\* now recur on right child \*/

printInorder(node->right);

}

/\* Driver code\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

// Function call

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

getchar();

return 0;

}

Pre-Order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node\* left;

struct node\* right;

};

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Given a binary tree, print its nodes in inorder\*/

void printInorder(struct node\* node)

{

if (node == NULL)

return;

/\* first recur on left child \*/

printInorder(node->left);

/\* then print the data of node \*/

printf("%d ", node->data);

/\* now recur on right child \*/

printInorder(node->right);

}

/\* Driver code\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

// Function call

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

getchar();

return 0;

}

Post –Order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node\* left;

struct node\* right;

};

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Given a binary tree, print its nodes according to the

"bottom-up" postorder traversal. \*/

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

// first recur on left subtree

printPostorder(node->left);

// then recur on right subtree

printPostorder(node->right);

// now deal with the node

printf("%d ", node->data);

}

/\* Driver code\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

// Function call

printf("\nPostorder traversal of binary tree is \n");

printPostorder(root);

getchar();

return 0;

}

Level Order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data,

pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node \*left, \*right;

};

/\* Function prototypes \*/

void printCurrentLevel(struct node\* root, int level);

int height(struct node\* node);

struct node\* newNode(int data);

/\* Function to print level order traversal a tree\*/

void printLevelOrder(struct node\* root)

{

int h = height(root);

int i;

for (i = 1; i <= h; i++)

printCurrentLevel(root, i);

}

/\* Print nodes at a current level \*/

void printCurrentLevel(struct node\* root, int level)

{

if (root == NULL)

return;

if (level == 1)

printf("%d ", root->data);

else if (level > 1) {

printCurrentLevel(root->left, level - 1);

printCurrentLevel(root->right, level - 1);

}

}

/\* Compute the "height" of a tree -- the number of

nodes along the longest path from the root node

down to the farthest leaf node.\*/

int height(struct node\* node)

{

if (node == NULL)

return 0;

else {

/\* compute the height of each subtree \*/

int lheight = height(node->left);

int rheight = height(node->right);

/\* use the larger one \*/

if (lheight > rheight)

return (lheight + 1);

else

return (rheight + 1);

}

}

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Driver program to test above functions\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

printf("Level Order traversal of binary tree is \n");

printLevelOrder(root);

return 0;

}

Diagonal Order

#include <bits/stdc++.h>

using namespace std;

// Tree node

struct Node

{

int data;

Node \*left, \*right;

};

/\* root - root of the binary tree

d - distance of current line from rightmost

-topmost slope.

diagonalPrint - multimap to store Diagonal

elements (Passed by Reference) \*/

void diagonalPrintUtil(Node\* root, int d,

map<int, vector<int>> &diagonalPrint)

{

// Base case

if (!root)

return;

// Store all nodes of same

// line together as a vector

diagonalPrint[d].push\_back(root->data);

// Increase the vertical

// distance if left child

diagonalPrintUtil(root->left,

d + 1, diagonalPrint);

// Vertical distance remains

// same for right child

diagonalPrintUtil(root->right,

d, diagonalPrint);

}

// Print diagonal traversal

// of given binary tree

void diagonalPrint(Node\* root)

{

// create a map of vectors

// to store Diagonal elements

map<int, vector<int> > diagonalPrint;

diagonalPrintUtil(root, 0, diagonalPrint);

cout << "Diagonal Traversal of binary tree : \n";

for (auto it :diagonalPrint)

{

vector<int> v=it.second;

for(auto it:v)

cout<<it<<" ";

cout<<endl;

}

}

// Utility method to create a new node

Node\* newNode(int data)

{

Node\* node = new Node;

node->data = data;

node->left = node->right = NULL;

return node;

}

// Driver program

int main()

{

Node\* root = newNode(8);

root->left = newNode(3);

root->right = newNode(10);

root->left->left = newNode(1);

root->left->right = newNode(6);

root->right->right = newNode(14);

root->right->right->left = newNode(13);

root->left->right->left = newNode(4);

root->left->right->right = newNode(7);

/\* Node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(9);

root->left->right = newNode(6);

root->right->left = newNode(4);

root->right->right = newNode(5);

root->right->left->right = newNode(7);

root->right->left->left = newNode(12);

root->left->right->left = newNode(11);

root->left->left->right = newNode(10);\*/

diagonalPrint(root);

return 0;

}

**LAB MST**

**Question Paper (Mapped with COs)**

**Scheme of Evaluation**

1. Set of Unique questions need to be prepared and distributed among the students.

2. Students need to take their seats according to their UIDs in Sequential fashion.

3. No student can change their allocated seat during the practical, to ensure this, you need to make all the computers in working mode. Ask the respective Lab Instructors to install the required Operating System/ Application Software in your guidance.

4. Faculty will visit to respective student desk for the evaluating the Lab components, as mentioned in the bullet 5.

5. Lab Evaluation Components are

a) Worksheet - 5 Marks

b) Conduct - 5 Marks

c) Viva - 10 Marks

Total Marks: 20

**Sample Answer Sheets**

**CO-Wise Mark Sheets**

Group A

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Student UID** | **Student Name** | **Conduct (CO1)** | **Worksheet (CO3)** | **Viva-voce (CO2)** | **Total (20 Marks)** |
| **(05 marks)** | **(05 marks)** | **(10 marks)** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

Group B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Student UID** | **Student Name** | **Conduct (CO1)** | **Worksheet (CO3)** | **Viva-voce (CO2)** | **Total (20 Marks)** |
| **(05 marks)** | **(05 marks)** | **(10 marks)** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

**List of Slow Learners**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UID** | **Name** | **MarksObtd\_Percentage** | **MarksObtd** | **MaxMarks** |
|  |  |  |  |  |
|  |  |  |  |  |

**Action Taken**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Schedule for Remedial Classes  Aug-Dec 2022** | | | | | | |
| **S.No** | **Day** | **Subject CODE** | **Subject Name** | **Room No** | **Faculty Name** | **Timing** |
|  |  |  |  |  |  |  |

**Best Practices Adopted**

1. Introductory first session about the subject followed by Getting Started Quiz.
2. Post Chapter Discussion Forums.
3. Real World to The Work /Instructor Lead Hands on Learning through Lab work.
4. Self-Assessment Questionnaires. (Through Assignment Task)
5. Peer Teaching
6. Promote Student Engagement through discussions and presentations.
7. Case Studies

**Final Award Sheet**

|  |  |  |  |
| --- | --- | --- | --- |
| **UID** | **CO2 MM:10** | **CO3 MM:15** | **CO4 MM:15** |
|  |  |  |  |

**List of Slow Learners**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UID** | **Name** | **MarksObtd\_Percentage** | **MarksObtd** | **MaxMarks** |
|  |  |  |  |  |

**Overall Observation and Action Taken (Recommendation to BOS)**