**DS- Exam 03 (Mid-Term) Answer Script**

|  |
| --- |
| Question No. 01` |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int data;      treeNode \*leftChild;      treeNode \*rightChild;      // constructor      treeNode(int value)      {          data = value;          leftChild = NULL;          rightChild = NULL;      }  };  // is similar two trees?  bool isSame(treeNode \*root1, treeNode \*root2)  {      bool x, y, z;      // NULL      if (root1 == NULL && root2 == NULL)      {          return true;      }      // not null      else if (root1 != NULL && root2 != NULL)      {          x = (root1->data == root2->data);          y = isSame(root1->leftChild, root2->leftChild);          z = isSame(root1->rightChild, root2->rightChild);      }      int res = x && y && z;      if (res == 1)      {          // cout << "res " << res << endl;          return true;      }      else      {          // cout << "res " << res << endl;          return false;      }  }  int main()  {      // first tree      // creat root node      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presentRoot = q.front(); // STL queue          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presentRoot->leftChild = n1;          presentRoot->rightChild = n2;          if (n1 != NULL)              q.push(n1);          if (n2 != NULL)              q.push(n2);      }      // second tree      int b;      cin >> b;      treeNode \*root2 = new treeNode(b);      queue<treeNode \*> q2;      q2.push(root2);      while (!q2.empty())      {          treeNode \*presentRoot2 = q2.front(); // STL queue          q2.pop();          int x2, y2;          cin >> x2 >> y2;          treeNode \*n3 = NULL;          treeNode \*n4 = NULL;          if (x2 != -1)          {              n3 = new treeNode(x2);          }          if (y2 != -1)          {              n4 = new treeNode(y2);          }          presentRoot2->leftChild = n3;          presentRoot2->rightChild = n4;          if (n3 != NULL)              q2.push(n3);          if (n4 != NULL)              q2.push(n4);      }      bool res = isSame(root, root2);      if (res == 1)      {          cout << "similar";      }      else      {          cout << "not similar";      }      return 0;  } |

|  |
| --- |
| Question No. 02 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  void level\_order(treeNode \*root)  {      if (root == NULL)      {          return;      }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              cout << presentNode->val << " ";              // levelOrder=levelOrder+to\_string(presentNode->val);              // levelOrder=levelOrder+" ";              if (presentNode->left != NULL)              {                  q.push(presentNode->left);              }              if (presentNode->right != NULL)              {                  q.push(presentNode->right);              }          }          else          {              if (!q.empty())              {                  q.push(NULL);              }          }      }  }  int main()  {      // root node      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesntNode = q.front();          q.pop();          // input child values          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesntNode->left = n1;          presesntNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      // level order traversal      // string levelOrder = "";      cout << "Lever order traversal: ";      level\_order(root);  } |

|  |
| --- |
| Question No. 03 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  void level\_order\_reverse(treeNode \*root)  {      if (root == NULL)      {          return;      }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      // stack for reverse      stack<int> s;      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              // cout << presentNode->val << " ";              s.push(presentNode->val);              if (presentNode->right != NULL) // right first              {                  q.push(presentNode->right);              }              if (presentNode->left != NULL)              {                  q.push(presentNode->left);              }          }          else          {              if (!q.empty())              {                  q.push(NULL); // level changes              }          }      }      cout << endl;      // stack      while (!s.empty())      {          cout << s.top() << " ";          s.pop();      }  }  int main()  {      // root node      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesntNode = q.front();          q.pop();          // input child values          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesntNode->left = n1;          presesntNode->right = n2;          if (n1 != NULL) // left          {              q.push(n1);          }          if (n2 != NULL) // right          {              q.push(n2);          }      }      cout<<"level order in reverse: "<<endl;      level\_order\_reverse(root);  } |

|  |
| --- |
| Question No. 04 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  bool symmatry(treeNode \*root1, treeNode \*root2)  {      if (root1 == NULL and root2 == NULL)      {          return true;      }      if (root1 != NULL and root2 != NULL)      {          if (root1->val == root2->val && symmatry(root1->left, root2->right) && symmatry(root1->right, root2->left))          {              return true;          }          else              return false;      }  }  bool isSymmetric(treeNode \*root)  {      return symmatry(root, root);  }  int main()  {      // root node      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesntNode = q.front();          q.pop();          // input child values          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesntNode->left = n1;          presesntNode->right = n2;          if (n1 != NULL) // left          {              q.push(n1);          }          if (n2 != NULL) // right          {              q.push(n2);          }      }      bool x = isSymmetric(root);      if (x == true)          cout << "symatric tree" << endl;      if (x == 0)          cout << "not symatric tree" << endl;  } |

|  |
| --- |
| Question No. 05 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  void zigzag\_order(treeNode \*root)  {      int count = 0;      if (root == NULL)      {          return;      }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      queue<int>z;      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              //cout << presentNode->val << " ";              z.push(presentNode->val);              // count even              if (count % 2 != 0)              {                  if (presentNode->left != NULL) //left first                  {                      q.push(presentNode->left);                  }                  if (presentNode->right != NULL)                  {                      q.push(presentNode->right);                  }              }              // count odd              if(count % 2 == 0)              {                  if (presentNode->right != NULL)//right first                  {                      q.push(presentNode->right);                  }                  if (presentNode->left != NULL)                  {                      q.push(presentNode->left);                  }                }          }          else          {              if (!q.empty())              {                  q.push(NULL);                  count++;              }          }      }      //cout << "count: " << count << endl;      while(!z.empty()){          cout<<z.front()<<" ";          z.pop();      }  }  int main()  {      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesentNode = q.front();          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesentNode->left = n1;          presesentNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      zigzag\_order(root);  } |

|  |
| --- |
| Question No. 06 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  void print(treeNode \*root)  {      if (root == NULL)      {          return;      }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      queue<int> r;      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              cout << presentNode->val << " ";              // r.push(presentNode->val);              if (presentNode->left != NULL)              {                  q.push(presentNode->left); // l              }              if (presentNode->right != NULL)              {                  q.push(presentNode->right); // r              }          }          else          {              if (!q.empty())              {                  q.push(NULL);              }          }      }      // while (!r.empty())      // {      //     cout << r.front() << " ";      //     r.pop();      // }  }  treeNode \*invert\_tree(treeNode \*root)  {      if (root == NULL)      {          return root;      }      // swap left-right      treeNode \*temp = root->left;      root->left = root->right;      root->right = temp;      // recursive call      invert\_tree(root->left);      invert\_tree(root->right);      return root;  }  int main()  {      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesentNode = q.front();          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesentNode->left = n1;          presesentNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      treeNode \*root2 = invert\_tree(root);      //check      print(root2);  } |

|  |
| --- |
| Question No. 07 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  int tilt = 0;  int findSum(treeNode \*root)  {      // following post order traversal system:left,root,right      if (root == NULL)          return 0;      int l = findSum(root->left);      int r = findSum(root->right);      tilt = tilt + abs(l - r); // node: left-right      return l + r + root->val; // returning to upper node  }  int findTilt(treeNode \*root)  {      if (root == NULL)          return 0;      findSum(root);      return tilt;  }  int main()  {      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesentNode = q.front();          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesentNode->left = n1;          presesentNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      int x = findTilt(root);      cout << x << endl;  } |

|  |
| --- |
| Question No. 08 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  void average\_level(treeNode \*root)  {      int sum = 0;      int count = 0;      if (root == NULL)      {          return;      }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              sum = sum + presentNode->val; //sum values of same level              count++;              //  cout << presentNode->val << " ";              if (presentNode->left != NULL) // left first              {                  q.push(presentNode->left);              }              if (presentNode->right != NULL)              {                  q.push(presentNode->right);              }          }          else          {              if (!q.empty())              {                  q.push(NULL); // level change              }              //print sum values after level change and then reset.              //cout << "elements: " << count << " sum: " << sum << endl;              double avg=(double)sum/(double)count;              cout<<avg<<" " ;              sum=0;              count=0;          }      }  }  int main()  {      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesentNode = q.front();          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesentNode->left = n1;          presesentNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      average\_level(root);  } |

|  |
| --- |
| Question No. 09 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  bool is\_unival(treeNode \*root)  {      if (root == NULL)      {          return true;      }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      queue<int> v;      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              // cout << presentNode->val << " ";              v.push(presentNode->val);              if (presentNode->left != NULL) // left first              {                  q.push(presentNode->left);              }              if (presentNode->right != NULL)              {                  q.push(presentNode->right);              }          }          else          {              if (!q.empty())              {                  q.push(NULL); // level change              }          }      }      int root\_val = v.front();      int flag = 1; // true      while (!v.empty())      {          if (root\_val != v.front())          {              flag = 0;              break;          }          else          {              flag = 1;          }          v.pop();      }      if(flag==0) return false; //not      else if(flag==1) return true; //yes      return -1;  }  int main()  {      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesentNode = q.front();          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesentNode->left = n1;          presesentNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      if (is\_unival(root))      {          cout << " uni-valued tree" << endl;      }      else      {          cout << " not uni-valued tree" << endl;      }  } |

|  |
| --- |
| Question No. 10 |
|  |
| Answer:  #include <bits/stdc++.h>  using namespace std;  class treeNode  {  public:      int val;      treeNode \*left;      treeNode \*right;      // constructor      treeNode(int val)      {          this->val = val;          left = NULL;          right = NULL;      }  };  int second\_minimum(treeNode \*root)  {      // if (root == NULL)      // {      //     return 1;      // }      queue<treeNode \*> q;      q.push(root);      q.push(NULL);      queue<int> r;      while (!q.empty())      {          treeNode \*presentNode = q.front();          q.pop();          if (presentNode != NULL)          {              // cout << presentNode->val << " ";              r.push(presentNode->val);              if (presentNode->left != NULL)              {                  q.push(presentNode->left); // l              }              if (presentNode->right != NULL)              {                  q.push(presentNode->right); // r              }          }          else          {              if (!q.empty())              {                  q.push(NULL);              }          }      }      int min\_first = root->val;      long long int x = LONG\_LONG\_MAX;//hypothetical      int res;      int flag = 0;      //cout << min\_first << " " << x << endl;      while (!r.empty())      {          res = r.front();          if (res > min\_first && res < x)          {              flag = 1;              x = res;              //cout << "x: " << x << endl;          }          r.pop();      }      if (flag == 1)      {          return x;      }      else      {          return -1;      }      // cout << x << endl;  }  int main()  {      int a;      cin >> a;      treeNode \*root = new treeNode(a);      queue<treeNode \*> q;      q.push(root);      while (!q.empty())      {          treeNode \*presesentNode = q.front();          q.pop();          int x, y;          cin >> x >> y;          treeNode \*n1 = NULL;          treeNode \*n2 = NULL;          if (x != -1)          {              n1 = new treeNode(x);          }          if (y != -1)          {              n2 = new treeNode(y);          }          presesentNode->left = n1;          presesentNode->right = n2;          if (n1 != NULL)          {              q.push(n1);          }          if (n2 != NULL)          {              q.push(n2);          }      }      int x = second\_minimum(root);      cout << x << endl;  } |

Submitted by: Subrata Saha

Email:[Subratabaec@gmail.com](mailto:Subratabaec@gmail.com)

Date: 03.09.2022

THE END