**Department of Information Technology**

**Data Structures (IT-201)**



Assignment 1

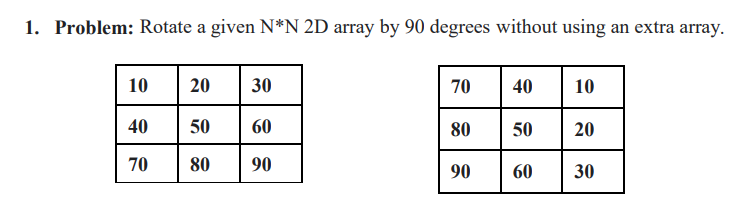
Submitted to-

Dr. Virender Ranga

Submitted by-

Ritwij Kashyap

2K21/IT/147



Code:

void rotate(int \*\*matrix, int n)

{

  for (int i = 0; i < n / 2; i++)

  {

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

    {

      int temp = matrix[i][j];

      matrix[i][j] = matrix[n - 1 - j][i];

      matrix[n - 1 - j][i] = matrix[n - 1 - i][n - 1 - j];

      matrix[n - 1 - i][n - 1 - j] = matrix[j][n - 1 - i];

      matrix[j][n - 1 - i] = temp;

    }

  }

}

void print(int \*\*arr, int n)

{

  for (int i = 0; i < n; i++)

  {

    for (int j = 0; j < n; j++)

    {

      cout << arr[i][j] << " ";

    }

    cout << endl;

  }

}

int main()

{

  int n;

  cin >> n;

  int \*\*matrix = new int \*[n];

  for (int i = 0; i < n; i++)

  {

    matrix[i] = new int[n];

    for (int j = 0; j < n; j++)

    {

      cin >> matrix[i][j];

    }

  }

  cout << endl;

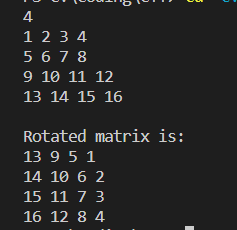
  cout << "Rotated matrix is:\n";

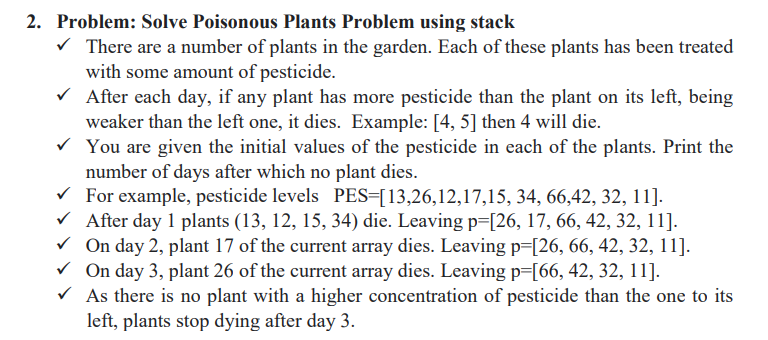
  rotate(matrix, n);

  print(matrix, n);

}

Output:





Code:

int main()

{

  stack<int> p1, p2;

  int choice = 1;

  while (choice)

  {

    int n;

    cin >> n;

    p1.push(n);

    cout << "Enter more elements? (1 for yes, 0 for no)" << endl;

    cin >> choice;

  }

  int count = 0;

  bool flag = true;

  while (flag)

  {

    flag = false;

    while (!p1.empty())

    {

      int n1 = p1.top();

      p1.pop();

      if (!p1.empty())

      {

        int n2 = p1.top();

        p1.pop();

        if (n1 < n2)

        {

          flag = true;

          p1.push(n2);

        }

        else

        {

          p2.push(n1);

          p1.push(n2);

        }

      }

      else

      {

        p2.push(n1);

      }

    }

    while (!p2.empty())

    {

      p1.push(p2.top());

      p2.pop();

    }

    count++;

  }

  cout<<"Final stack of plants is:";

  while (!p1.empty())

  {

    cout<<p1.top()<<" ";

    p1.pop();

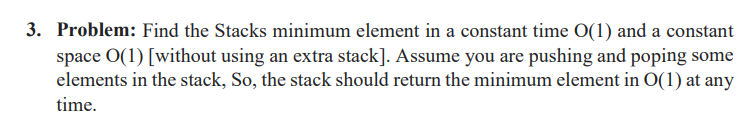
  }

  cout<<"\nNumber of days: "<<count-1;

}

Output:





Code:

struct Stack {

    stack<int> s;

    int minEle;

    void getMin()

    {

        if (s.empty())

            cout << "Stack is empty\n";

        else

            cout << "Minimum Element in the stack is: "

                 << minEle << "\n";

    }

    void peek()

    {

        if (s.empty()) {

            cout << "Stack is empty ";

            return;

        }

        int t = s.top();

        cout << "Top Most Element is: ";

        (t < minEle) ? cout << minEle : cout << t;

    }

    void pop()

    {

        if (s.empty()) {

            cout << "Stack is empty\n";

            return;

        }

        cout << "Top Most Element Removed: ";

        int t = s.top();

        s.pop();

        if (t < minEle) {

            cout << minEle << "\n";

            minEle = 2 \* minEle - t;

        }

        else

            cout << t << "\n";

    }

    void push(int x)

    {

        if (s.empty()) {

            minEle = x;

            s.push(x);

            cout << "Number Inserted: " << x << "\n";

            return;

        }

        else if (x < minEle) {

            s.push(2 \* x - minEle);

            minEle = x;

        }

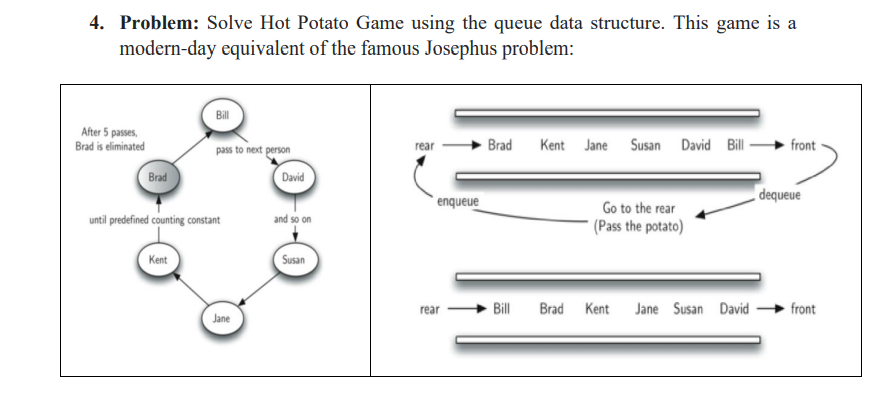
        else

            s.push(x);

        cout << "Number Inserted: " << x << "\n";

    }

};



Code:

void lastSurvivor(queue<string>\* line, int n)

{

    while (line->size()!=1)

    {

        for (int i = 0; i < n; i++)

        {

            line->push(line->front());

            line->pop();

        }

        cout<<"Eliminated: "<<line->front()<<endl;

        line->pop();

    }

}

int main()

{

    queue<string>\* line=new queue<string>;

    line->push("Bill");

    line->push("David");

    line->push("Susan");

    line->push("Jane");

    line->push("Kent");

    line->push("Brad");

    int n;

    cout<<"Enter number of passes:"<<endl;

    cin>>n;

    lastSurvivor(line,n);

    cout<<"The last survivor is: "<<line->front();

}

Output:

