```c
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

struct Node{
    float data;
    struct Node * next;
};

struct Node* top = NULL;

void linkedListTraversal(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}

int isEmpty(struct Node* top){
    if (top==NULL){
        return 1;
    }
    else{
        return 0;
    }
}

int isFull(struct Node* top){
    struct Node* p = (struct Node*)malloc(sizeof(struct Node));
    if(p==NULL){
        return 1;
    }
    else{
        return 0;
    }
}

```

```
40: struct Node* push(struct Node* top, float x){
41:     if(isFull(top)){
42:         printf("Stack Overflow\n");
43:     }
44:     else{
45:         struct Node* n = (struct Node*) malloc(sizeof(struct Node)
46:         n->data = x;
47:         n->next = top;
48:         top = n;
49:         return top;
50:     }
51: }
52:
53: int pop(struct Node*tp){
54:     if(isEmpty(tp)){
55:         printf("Stack Underflow\n");
56:     }
57:     else{
58:         struct Node* n = tp;
59:         top = (tp)->next;
60:         int x = n->data;
61:         free(n);
62:         return x;
63:     }
64: }
65:
66:
67:
68: float determinent(float matrix[25][25], float size)
69:
70: {
71:
72:     int c;
73:     float det=0,s=1;
74:     float b[25][25];
75:     int i,j;
76:     int m,n;
77:     if(size == 1){
78:       return (matrix[0][0]);}
```

```
    else{
        det=0;
        for(c=0; c<size; c++){
            m=0;
            n=0;
            for(i=0; i<size; i++){
                for(j=0; j<size; j++){
                    b[i][j] = 0;
                    if(i!=0 && j!=c){
                        b[m][n] = matrix[i][j];
                        if(n<(size-2)){
                            n++;
                        }else{
                             n=0;
                             m++;
                             }
                    }
                }
            }
            det = det + s*(matrix[0][c]*determinent(b,size-1));
            s = -1*s;
        }
    }
    return det;
}

int main(){

    float k;
    printf("Enter the size n*n of the matrix ");
    scanf("%f",&k);

    int i,j;
    float matrix[25][25];

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

```
118:             printf("Enter the %d%d element of the matrix ",i,j);
119:             scanf("%f", &matrix[i][j]);
120:         }
121:     }
122:     float result=determinent(matrix,k);
123:     printf("\nThe determinant of the matrix is %f",result);
124:
125:
126:     float cofactor[25][25];
127:
128:     if(result == 0)
129:         printf("\nMatrix is singular, the inverse of the matrix do
130:     else
131:     {
132:         int c,d,p,q;
133:         int m,n;
134:         int size = k;
135:         float b[25][25];
136:         for(c=0; c<size; c++)
137:         {
138:             for(d=0; d<size; d++)
139:             {
140:                 m=0;
141:                 n=0;
142:                 for(p=0; p<size; p++)
143:                 {
144:                     for(q=0; q<size; q++)
145:                     {
146:                         if(p!=c && q!=d)
147:                         {
148:                             b[m][n] = matrix[p][q];
149:                             if(n<(size-2))
150:                             {
151:                                 n++;
152:                             }
153:                             else
154:                             {
155:                                 n=0;
156:                                 m++;
```

```
157:                             }
158:                         }
159:
160:                 top= push(top,pow(-1,(c+d))*determinent(b,k-1));
161:                 float ram=pop(top);
162:                 cofactor[c][d]=ram;
163:                 printf("Cofactor is %f ",ram);
164:                 printf("\n");
165:                     }
166:                 }
167:            }
168:          }
169:
170:         float Adjoint[25][25];
171:
172:         int s,t;
173:         for(s=0; s<k; s++)
174:         {
175:             printf("\n");
176:             for(t=0; t<k; t++)
177:             {
178:                 top=push(top,cofactor[t][s]);
179:                 float w=pop(top);
180:                 Adjoint[s][t]=w;
181:                 printf("Adjoint is %f",w);
182:                 printf("\n");
183:             }
184:         }
185:
186:         float Inverse[25][25];
187:
188:         int l,z;
189:         for(l=0; l<k; l++)
190:         {
191:             for(z=0; z<k; z++)
192:             {
193:                 top =push(top,(Adjoint[l][z]/result));
194:                 float vivu=pop(top);
195:                 printf("Value in Computation of Inverse of the mat
```

```
196:                 printf("\n");
197:             }
198:         }
199:         int e,f;
200:         printf("The inverse of the matrix is");
201:
202:         for(e=0; e<k; e++)
203:         {
204:             printf("\n");
205:             for(f=0; f<k; f++)
206:             {
207:                 top =push(top,(Adjoint[e][f]/result));
208:                 float satya=pop(top);
209:                 Inverse[e][f]=satya;
210:                 printf("%f ",satya);
211:             }
212:         }
213:     }
214:     printf("\n");
215:     return 0;
216: }
```