Implementation of BSF

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

int adjacancy[10][10];
int queue[20], front = 0, rear = -1;
int visited[20];
int n, i, j;

void bfs(int startVertex) {
   for (i = 1; i <= n; i++) {
      if (adjacancy[startVertex][i] && !visited[i]) {
         queue[++rear] = i;
      }
   }
   if (front <= rear) {
      visited[queue[front]] = 1;
      bfs(queue[front++]);
   }
}

int main() {
   printf("Enter number of vertices :  ");
   scanf("%d", &n);

   for (i = 1; i <= n; i++) {    // intialisation
      queue[i] = 0;
      visited[i] = 0;
   }

   // input of adjacancy matrix
   printf("\nEnter adjacancy matrix :  \n");
   for (i = 1; i <= n; i++) {
      for (j = 1; j <= n; j++) {
         scanf("%d", &adjacancy[i][j]);
      }
   }
   bfs(1);

   printf("\nBFS traversal is :  ");
   for (i = 1; i <= n; i++) {
      if (visited[i]) {
         printf("%d   ", i);
      }
      else {
         printf("\nEnetered graph is incorrect, since all nodes are not reachable!");
         break;
      }
   }
}
```

// output

```
Enter number of vertices :  5

Enter the adjacency matrix :
0 1 0 1 0
1 0 1 1 0
0 1 0 0 1
1 1 0 0 1
0 0 1 1 0
BFS Traversal :  A   B   D   C
```

Implementation of DFS

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

int n;
void dfs(int adj[][n],int visited[],int start) {
   int stack[n];
   int top = -1, i;
   printf("%c->  ",start+65);
   visited[start] = 1;
   stack[++top] = start;
   while(top != -1) {
      start = stack[top];
      for(i = 0; i < n; i++) {
         if(adj[start][i] && visited[i] == 0) {
            stack[++top] = i;
            printf("%c->  ", i+65);
            visited[i] = 1;
            break;
         }
      }
      if (i == n) {
         top--;
      }
   }
}

int main() {
   int adj[10][10];
   int visited[20] = {0}, i, j;

   printf("Enter number of vertices :  ");
   scanf("%d", &n);

   printf("\nEnter the adjacency matrix: \n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         scanf("%d", &adj[i][j]);
      }
   }
   printf("DFS Traversal :  ");
```

```
    dfs(adj,visited,0);
    return 0;
}

// output
  Enter number of vertices :  5

  Enter the adjacency matrix:
  0 1 0 1 0
  1 0 1 1 0
  0 1 0 0 1
  1 1 0 0 1
  0 0 1 1 0
  DFS Traversal :  A->  B->  C->  D->  E->
```