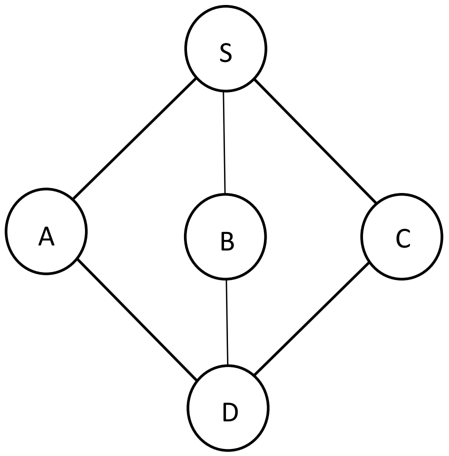
# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```



Queue

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |

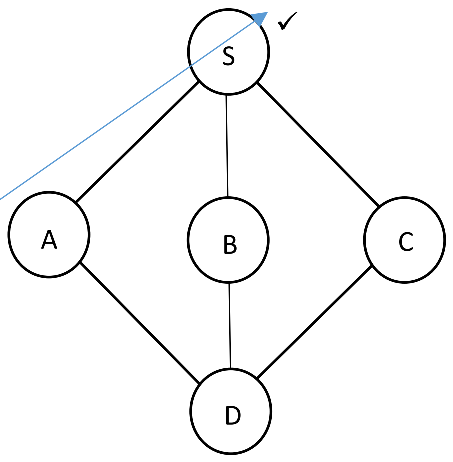
**OUTPUT**

# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```



Queue

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |

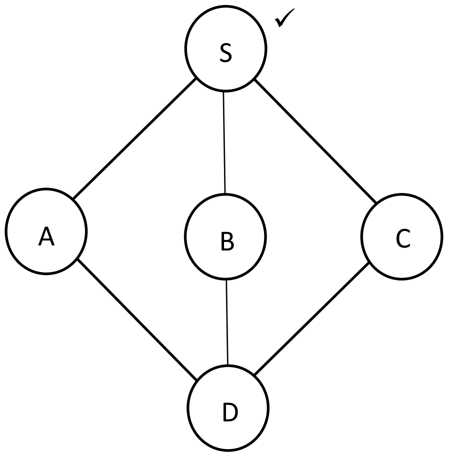
**OUTPUT**

BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```



Queue

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |   |   |

**OUTPUT**

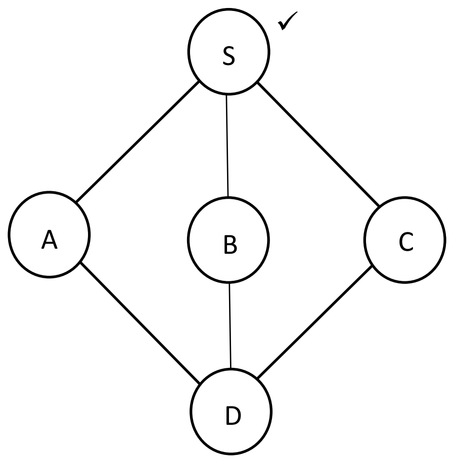
S

# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```



Queue

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 0 |   |   |   |   |   |   |   |   |

S

**OUTPUT**

S

# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

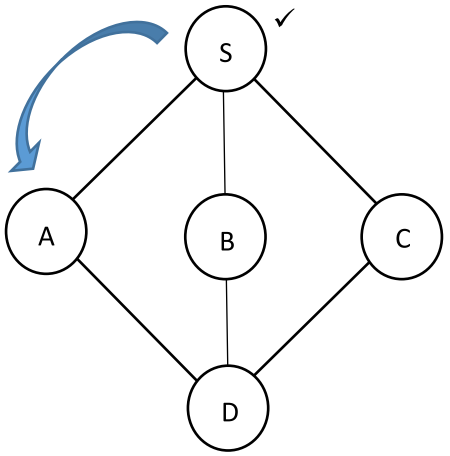
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 0

returns 1 (A)

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |

S

**OUTPUT**

S

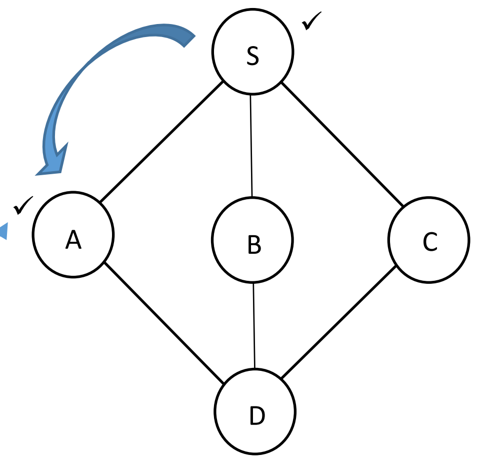
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Queue

| 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|

S

**OUTPUT**

S

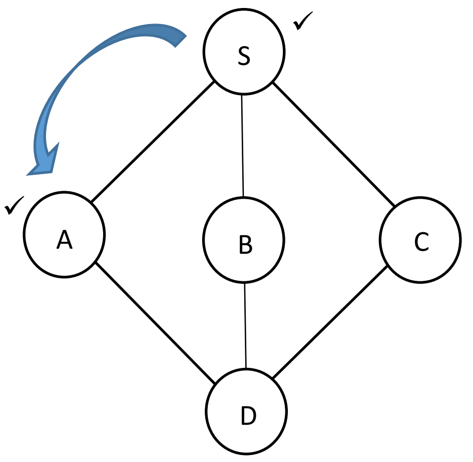
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Queue

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 0 |   |   |   |   |   |   |   |   |
| S |   |   |   |   |   |   |   |   |

**OUTPUT**

$S \rightarrow A$

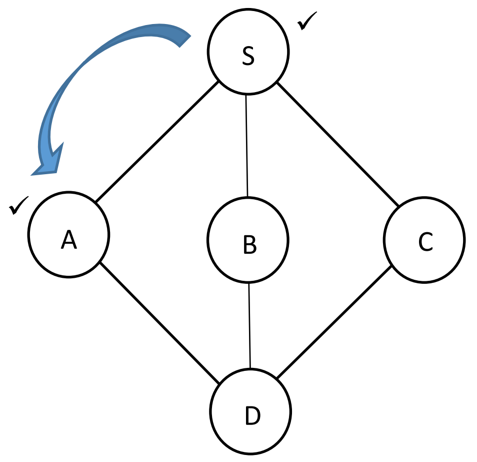
# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Queue

| 0 | 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | | | | | | | |

**OUTPUT**

S A

# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

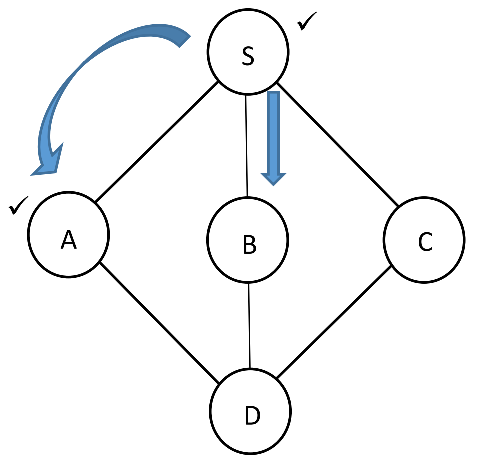
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 0

returns 2 (B)

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Queue

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 |   |   |   |   |   |   |   |
| S | A |   |   |   |   |   |   |   |

**OUTPUT**

S A

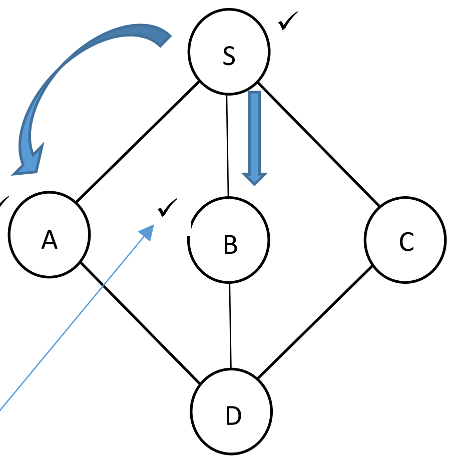
# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | | | |

S A

**OUTPUT**

S A

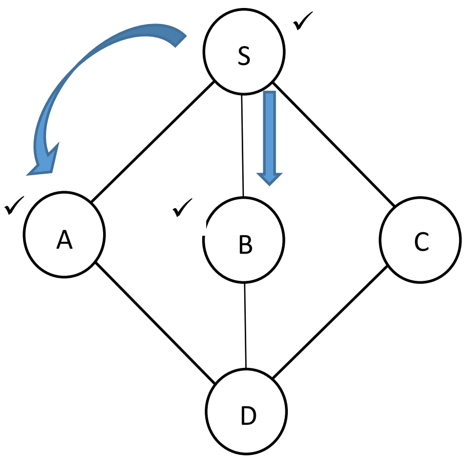
# BREADTH FIRST TRAVERSAL

```
                       0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | | | |
| S | A | | | | | | | |

**OUTPUT**

S A B

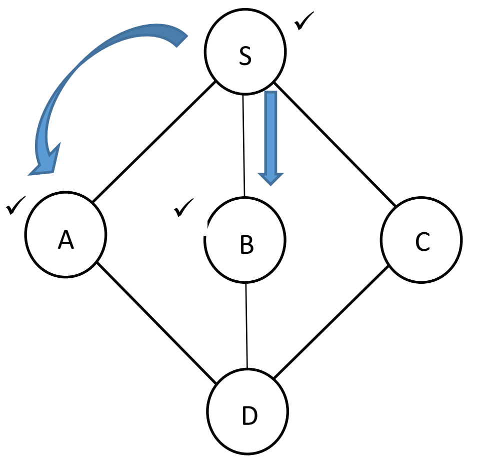
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Queue

| 0 | 1 | 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | B | | | | | | |

**OUTPUT**

S A B

# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

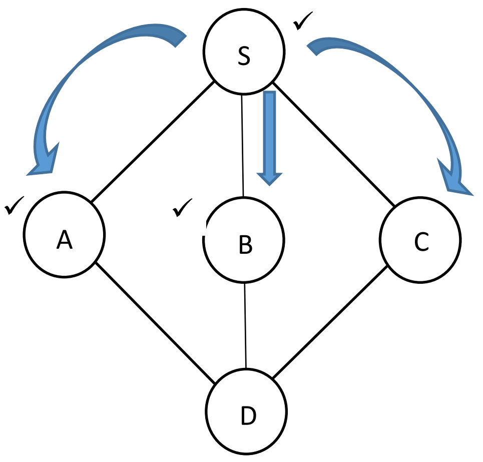
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 0

returns 3 (C)

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Queue

| 0 | 1 | 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | B | | | | | | |

**OUTPUT**

S A B

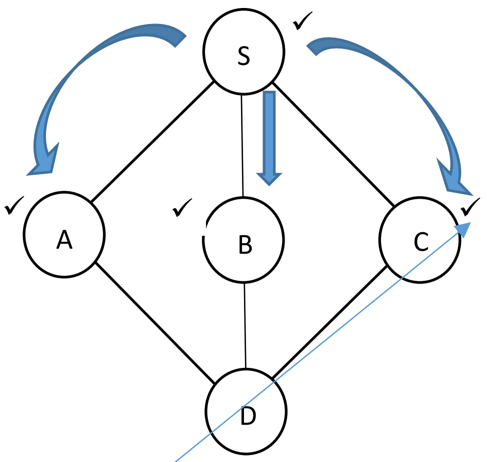
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | | | | | | |
| S | A | B | | | | | | |

**OUTPUT**

S A B

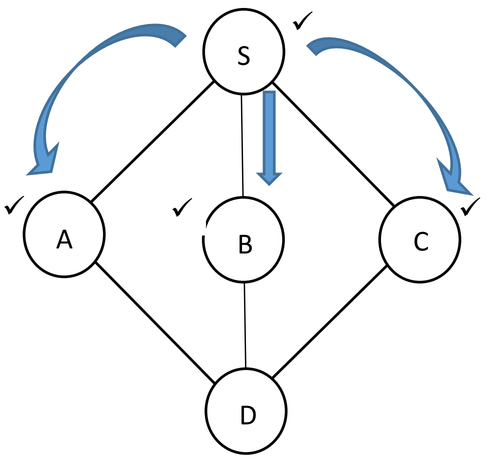
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | | | | | | |
| S | A | B | | | | | | |

**OUTPUT**

S A B C

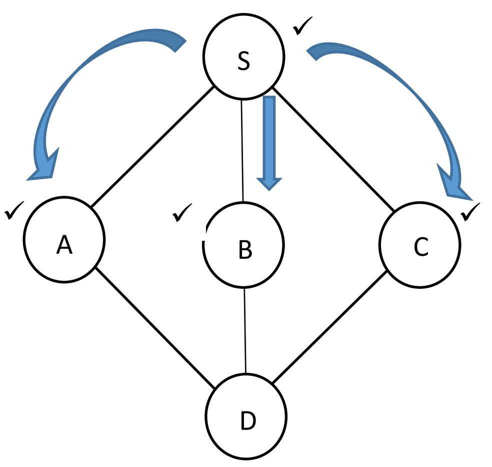
# BREADTH FIRST TRAVERSAL

```
                      0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

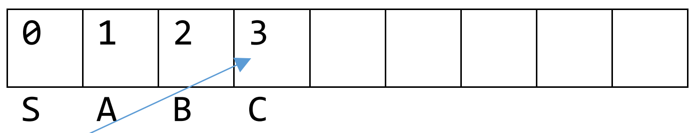
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Queue



**OUTPUT**

S A B C

BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

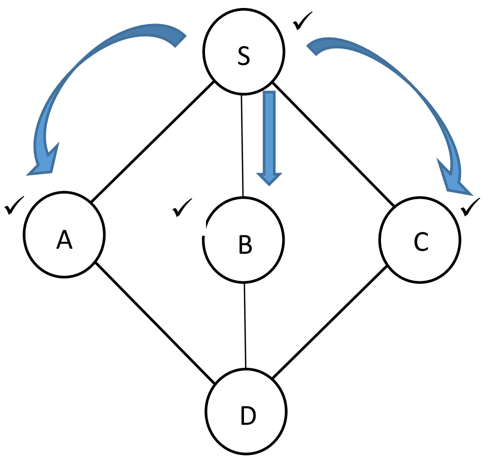
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 0

returns -1

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | | | | | |
| S | A | B | C | | | | | |

**OUTPUT**

S A B C

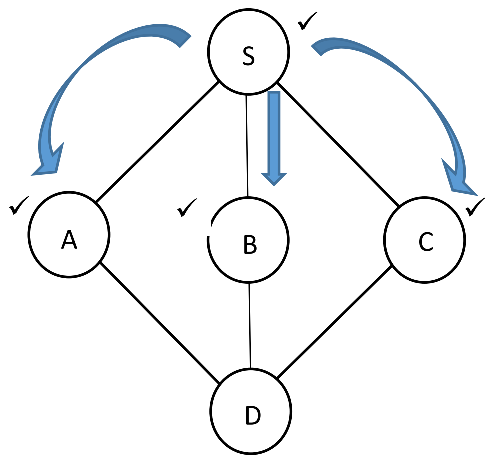
# BREADTH FIRST TRAVERSAL

```
                   0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

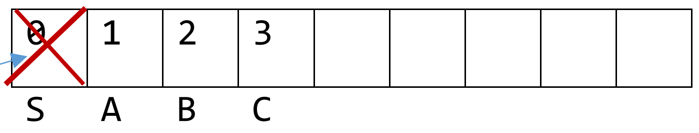
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue



**OUTPUT**

S A B C

# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

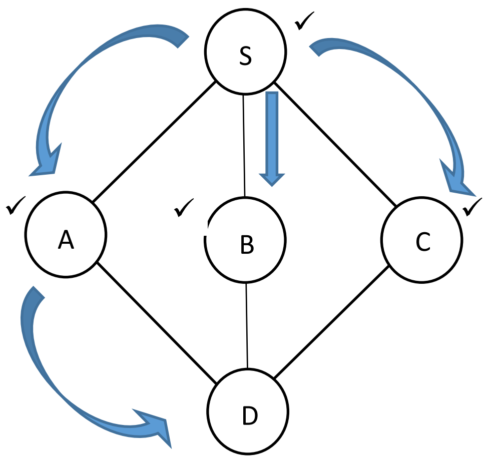
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 1

returns 4 (D)

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4



Queue

| ~~0~~ | 1 | 2 | 3 | | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | B | C | | | | | |

**OUTPUT**

S A B C

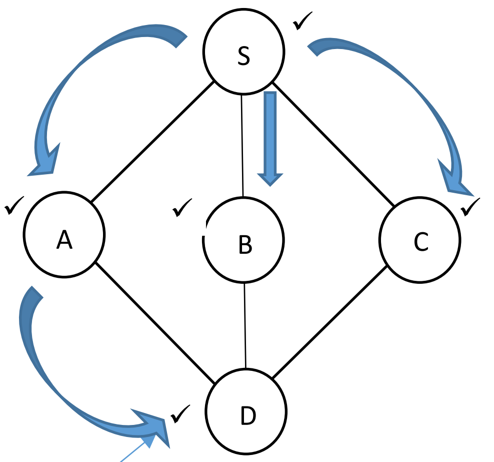
# BREADTH FIRST TRAVERSAL

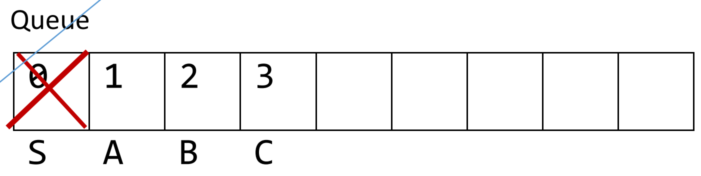
```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4





**OUTPUT**

S A B C

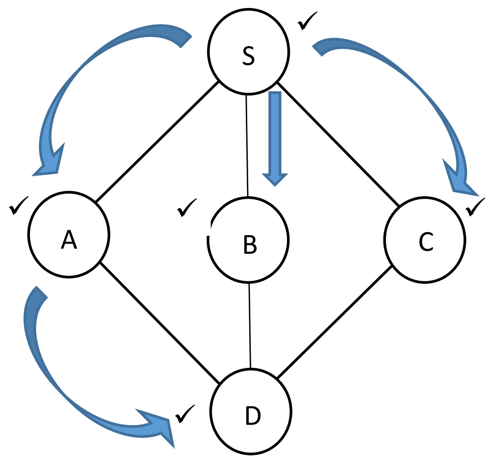
BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ~~0~~ | 1 | 2 | 3 | | | | | |
| S | A | B | C | | | | | |

**OUTPUT**

S A B C D

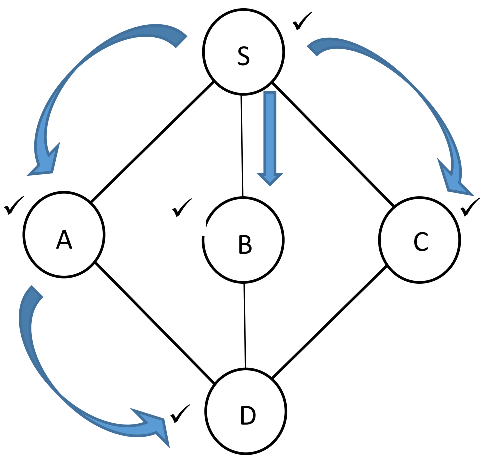
# BREADTH FIRST TRAVERSAL

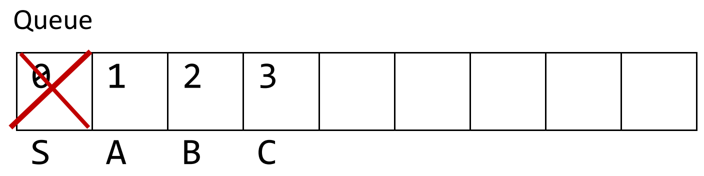
```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4





**OUTPUT**

S A B C D

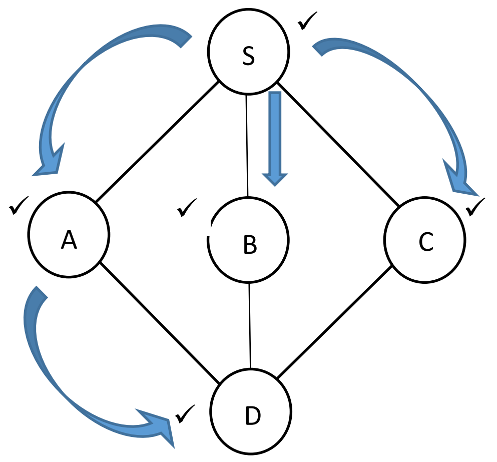
# BREADTH FIRST TRAVERSAL

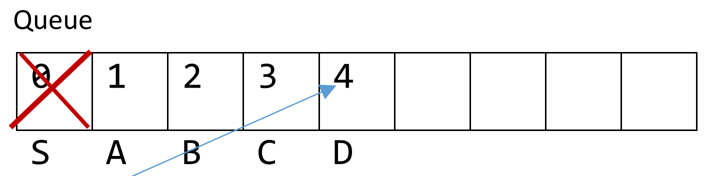
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```c
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```c
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4





**OUTPUT**

S A B C D

# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

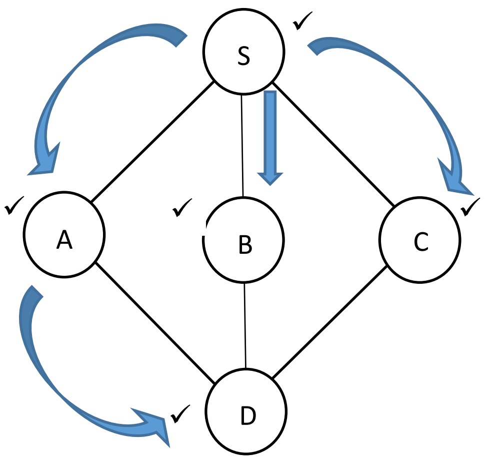
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 1

returns -1

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue

| ~~0~~ | 1 | 2 | 3 | 4 | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | B | C | D | | | | |

**OUTPUT**

S A B C D

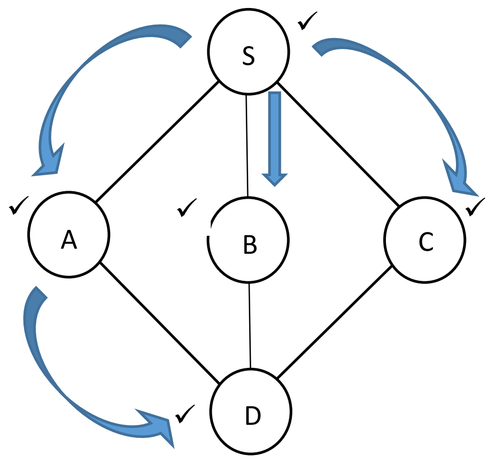
# BREADTH FIRST TRAVERSAL

```
                      0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue

| ~~0~~ | ~~1~~ | 2 | 3 | 4 | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | B | C | D | | | | |

**OUTPUT**

S A B C D

# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

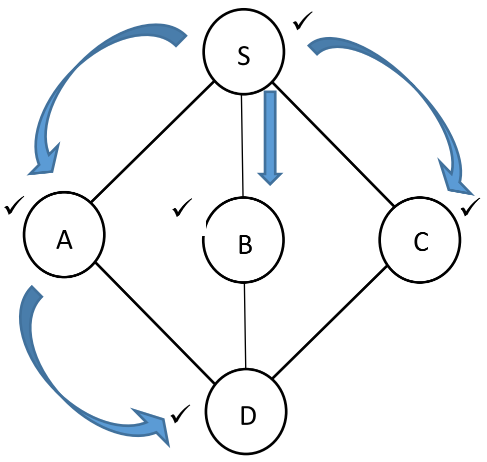
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 2

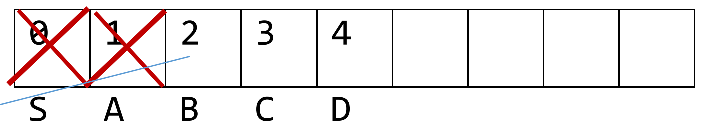
returns -1

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue



**OUTPUT**

S A B C D

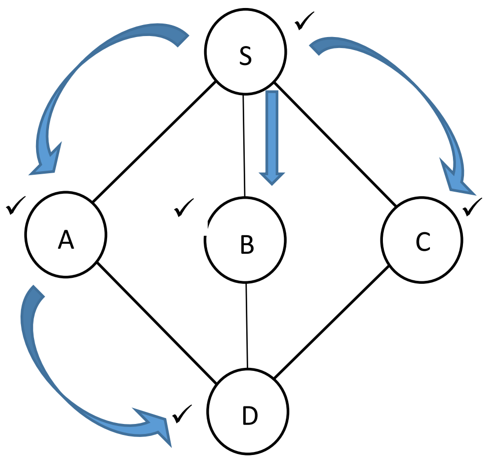
# BREADTH FIRST TRAVERSAL

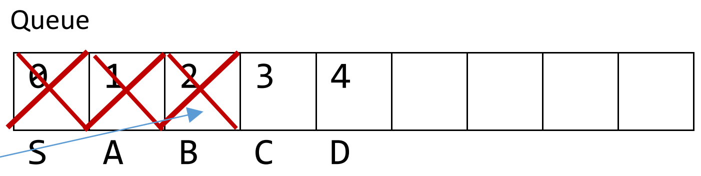
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1





**OUTPUT**

S A B C D

# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

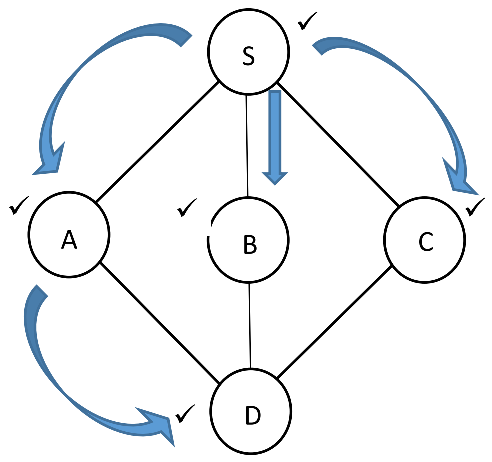
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 3

returns -1

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue

| ~~0~~ | ~~1~~ | ~~2~~ | 3 | 4 | | | | |
|---|---|---|---|---|---|---|---|---|
| S | A | B | C | D | | | | |

**OUTPUT**

S A B C D

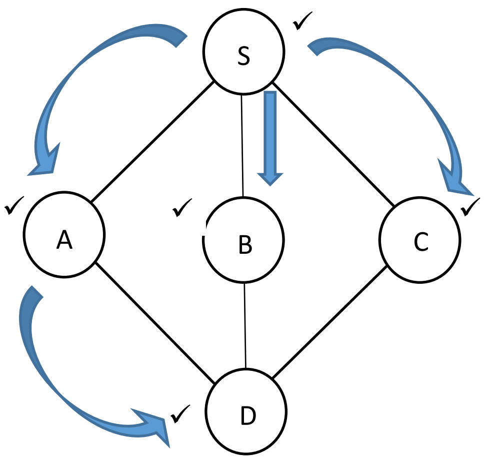
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

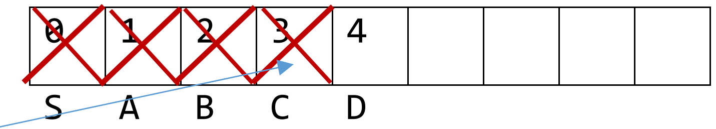
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue



**OUTPUT**

S A B C D

# BREADTH FIRST TRAVERSAL

```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

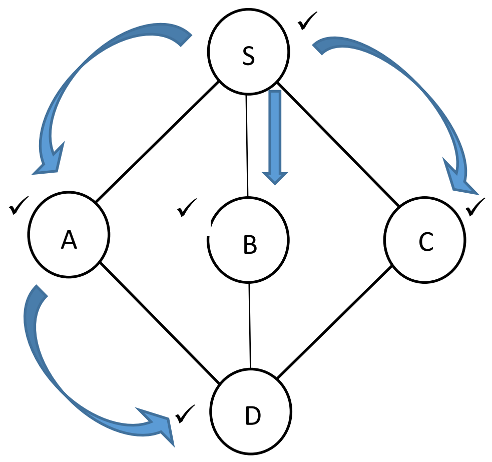
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 4

returns -1

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ~~0~~ | ~~1~~ | ~~2~~ | ~~3~~ | 4 | | | | |
| S | A | B | C | D | | | | |

**OUTPUT**

S A B C D

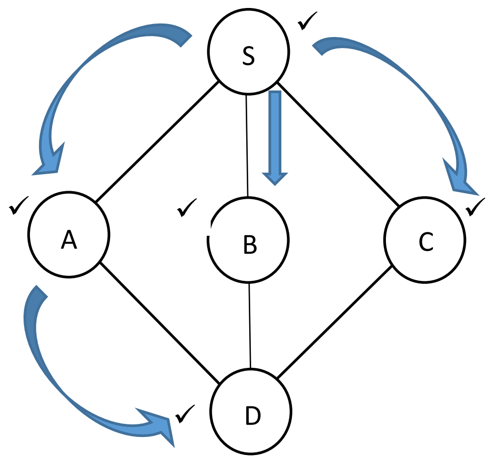
# BREADTH FIRST TRAVERSAL

```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

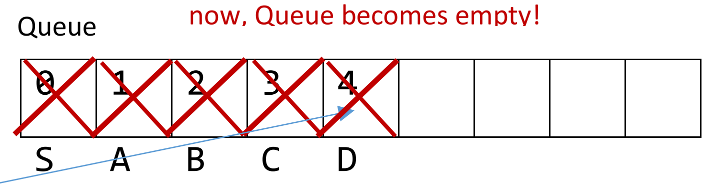
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



now, Queue becomes empty!



**OUTPUT**

S A B C D

# BREADTH FIRST TRAVERSAL

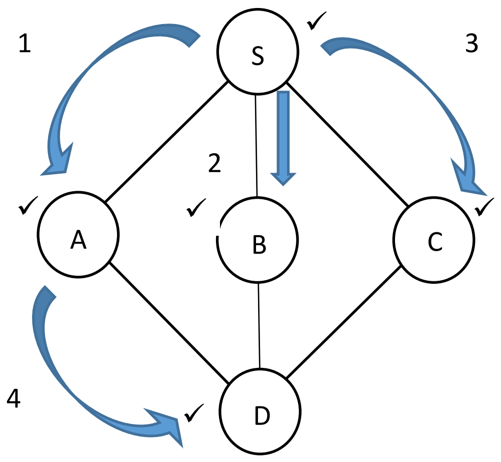
```
                     0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

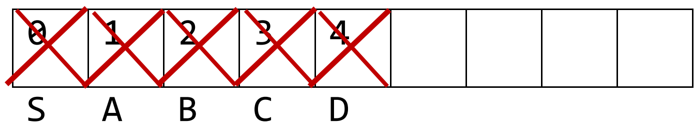
```
void breadthFirstTraversal() {
   int unvisitedVertex;
   queue_t Q;
   initializeQ(&Q);
   setUnvisited(listVertices, vertexCount);
   listVertices[0]->visited = true;
   printf("%c ", listVertices[0]->label);
   insert(&Q, 0);
   while (!isEmptyQ(&Q)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(Q));
      if (unvisitedVertex == -1)
         remove(&Q);
      else {
         listVertices[unvisitedVertex]->visited = true;
         printf("%c ", listVertices[unvisitedVertex]->label);
         insert(&Q, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1

End of function



Queue is empty!



**OUTPUT**

S A B C D