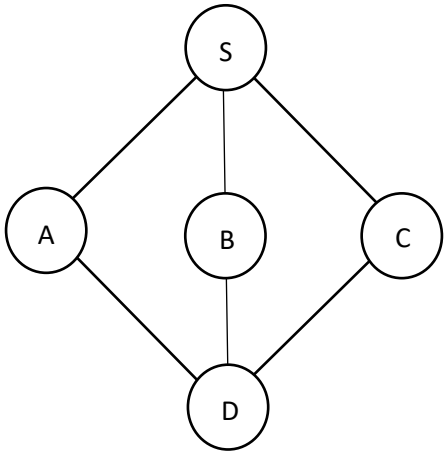
DEPTH FIRST TRAVERSAL

```
                    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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack

top

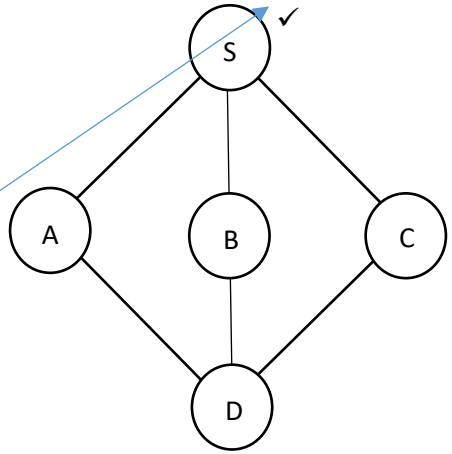
OUTPUT

DEPTH 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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack

top

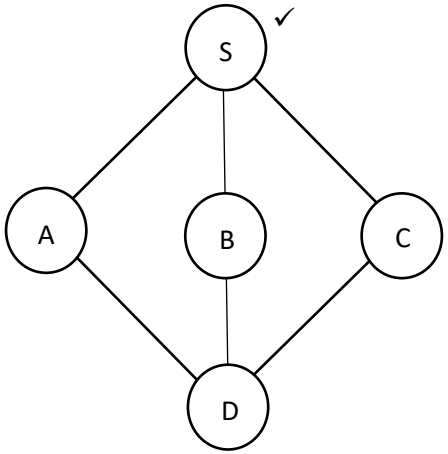
OUTPUT

DEPTH 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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack

top

OUTPUT

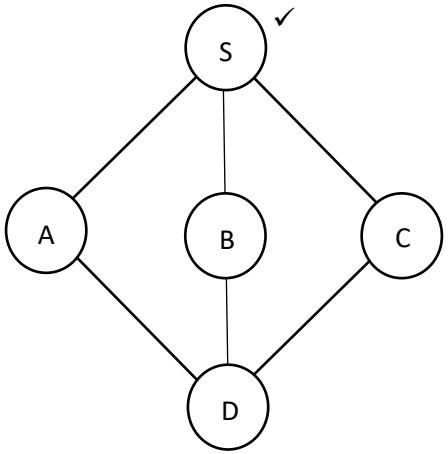
S

DEPTH FIRST TRAVERSAL

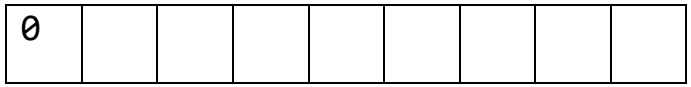
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack



S

top

OUTPUT

S

DEPTH 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 |
```

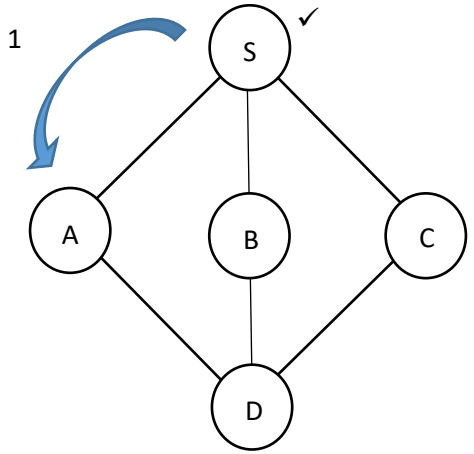
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 0

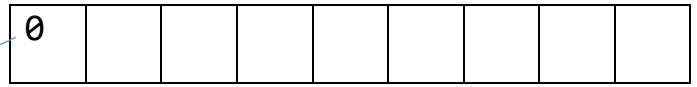
returns 1 (A)

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Stack



S

top

OUTPUT

S

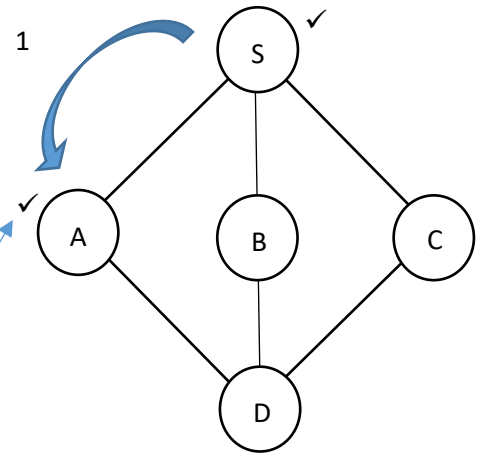
DEPTH FIRST TRAVERSAL

```
                    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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Stack

S

top

OUTPUT

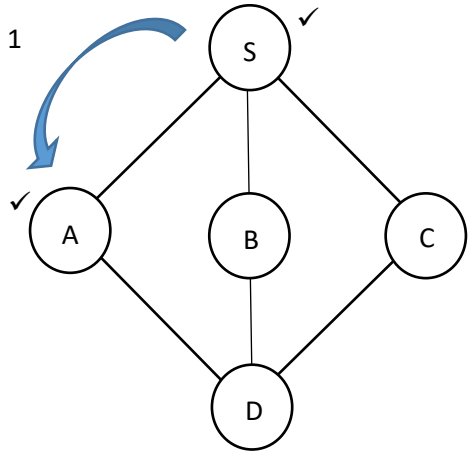
S

DEPTH FIRST TRAVERSAL

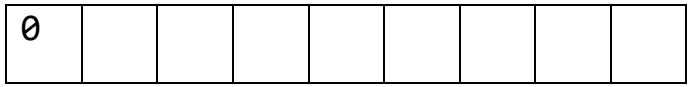
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                                  unvisitedVertex=1
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack



S

top

OUTPUT

S A

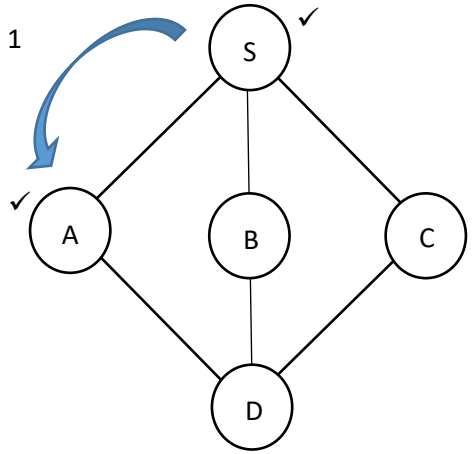
DEPTH FIRST TRAVERSAL

```
                    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 |
```

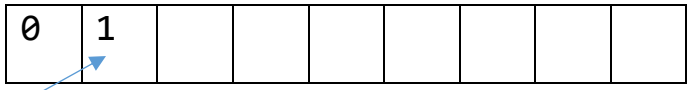
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=1



Stack



OUTPUT

S A

DEPTH 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 |
```

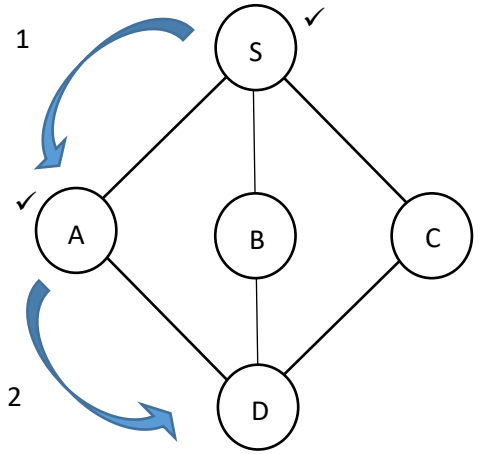
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 1

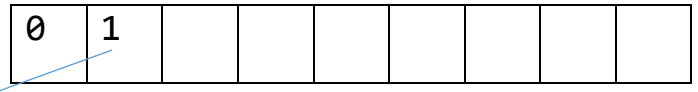
returns 4 (D)

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4



Stack



OUTPUT

S A

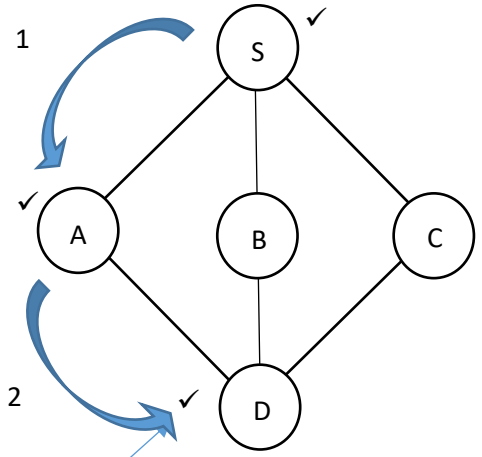
DEPTH FIRST TRAVERSAL

```
                    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 |
```

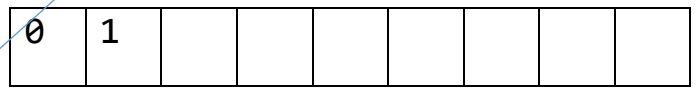
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4



Stack



OUTPUT

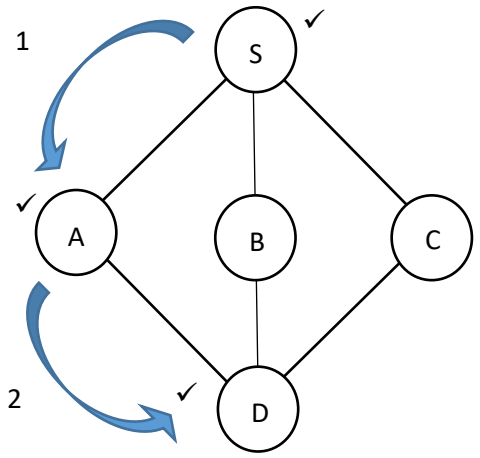
S A

DEPTH 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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                                  unvisitedVertex=4
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack

S A

top

OUTPUT

S A D

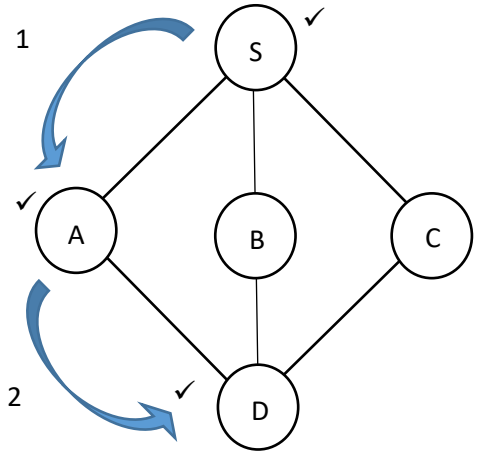
DEPTH FIRST TRAVERSAL

```
                    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 |
```

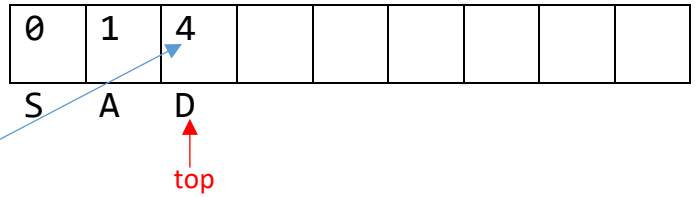
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=4



Stack



OUTPUT

S A D

DEPTH 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 |
```

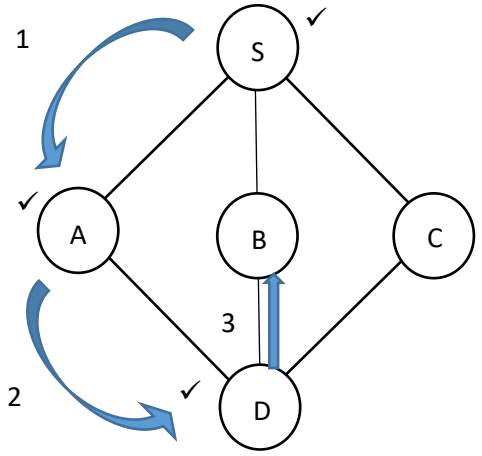
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 4

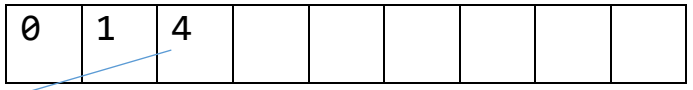
returns 2 (B)

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Stack



OUTPUT

S A D

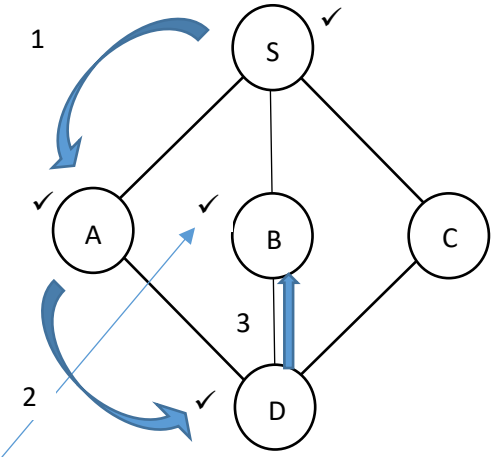
DEPTH FIRST TRAVERSAL

```
                    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 |
```

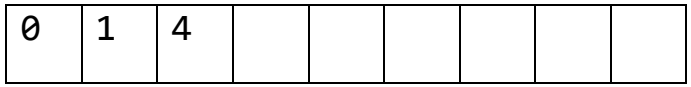
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Stack



OUTPUT

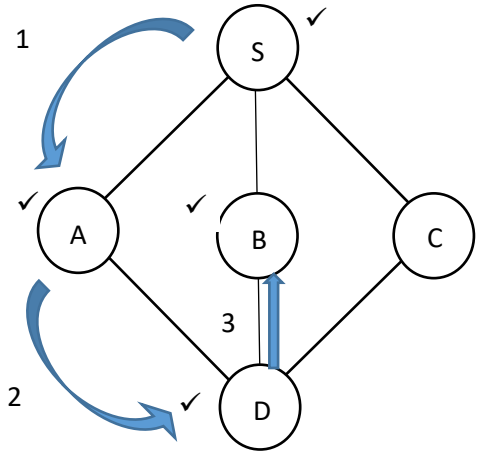
S A D

DEPTH FIRST TRAVERSAL

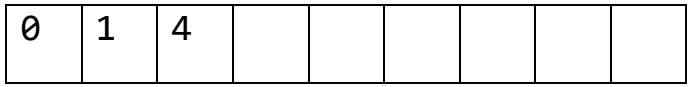
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                                  unvisitedVertex=2
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack



OUTPUT

S A D B

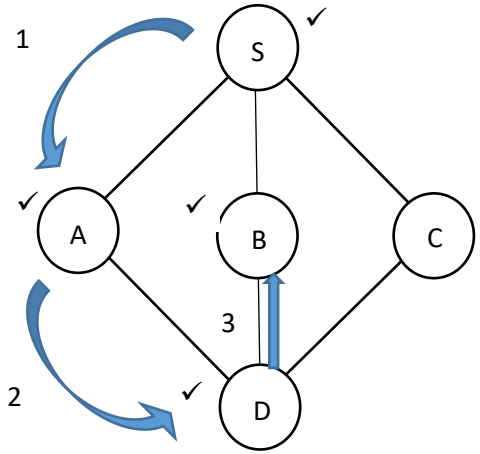
DEPTH FIRST TRAVERSAL

```
                    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 |
```

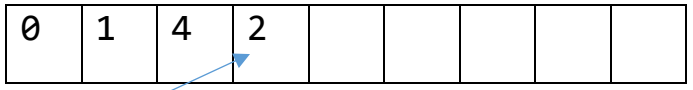
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=2



Stack



OUTPUT

S A D B

DEPTH 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 |
```

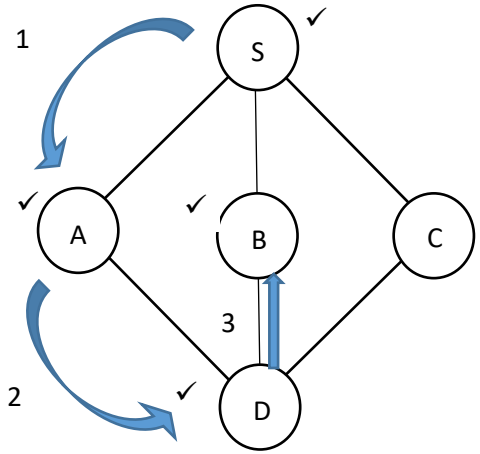
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 2

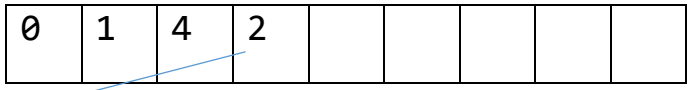
returns -1

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Stack



OUTPUT

S A D B

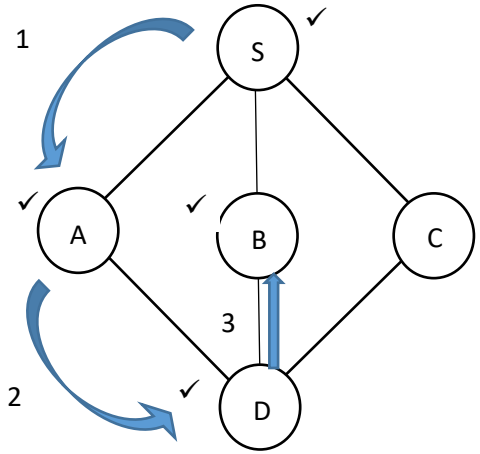
DEPTH FIRST TRAVERSAL

```
                    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 |
```

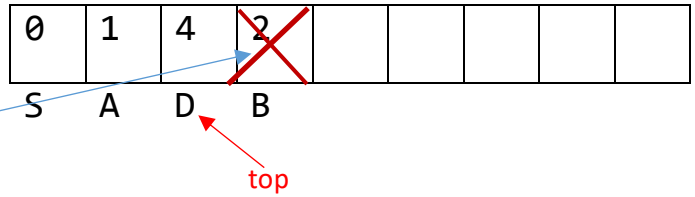
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Stack



OUTPUT

S A D B

DEPTH 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 |
```

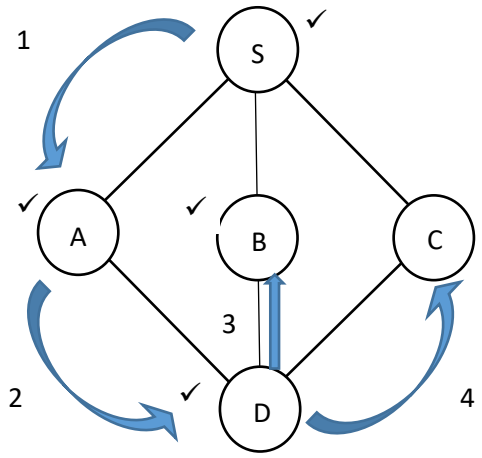
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 4

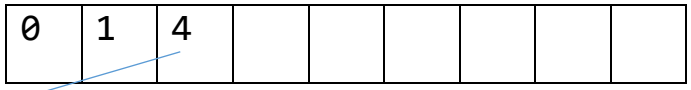
returns 3

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Stack



OUTPUT

S A D B

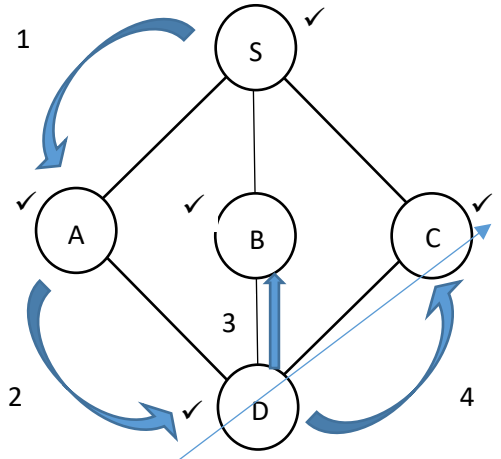
DEPTH FIRST TRAVERSAL

```
                    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 |
```

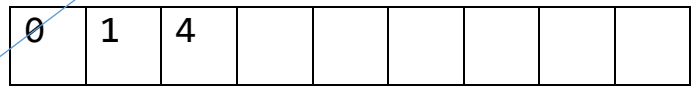
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Stack



OUTPUT

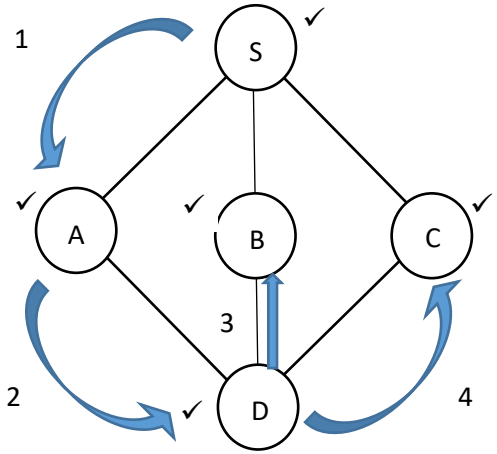
S A D B

DEPTH 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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                                  unvisitedVertex=3
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack

S A D

top

OUTPUT

S A D B C

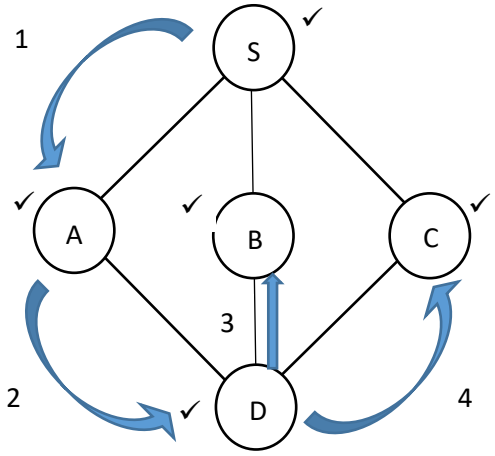
DEPTH FIRST TRAVERSAL

```
                    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 |
```

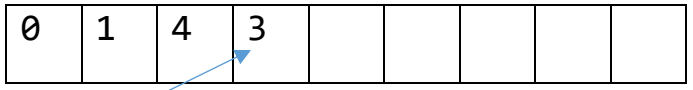
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=3



Stack



OUTPUT

S A D B C

DEPTH 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 |
```

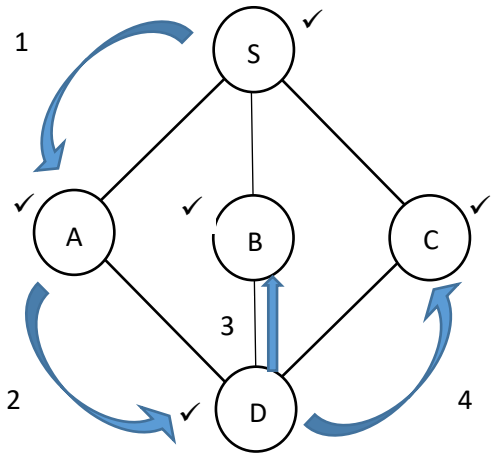
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 3

returns -1

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Stack

0	1	4	3					
S	A	D	C					

top

OUTPUT

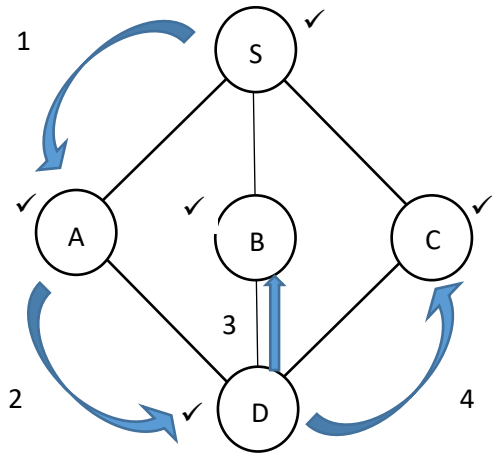
S A D B C

DEPTH FIRST TRAVERSAL

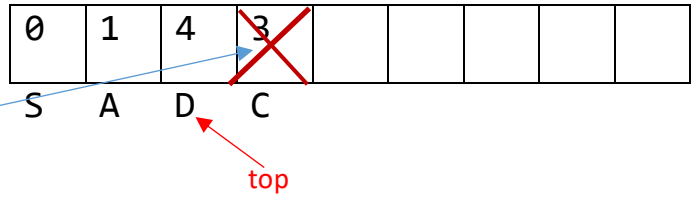
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                          unvisitedVertex=-1
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack



OUTPUT

S A D B C

DEPTH 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 |
```

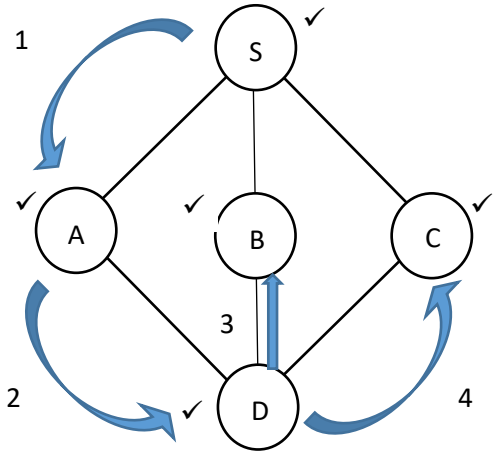
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 4

returns -1

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Stack

0	1	4						
S	A	D						

top

OUTPUT

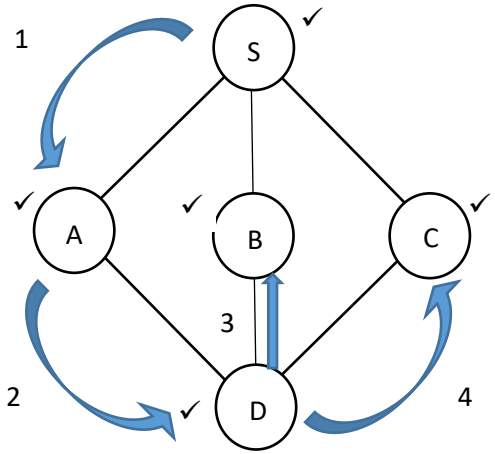
S A D B C

DEPTH FIRST TRAVERSAL

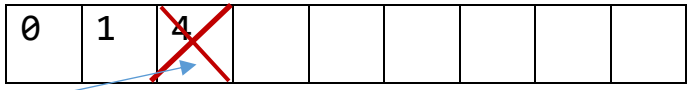
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                          unvisitedVertex=-1
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```



Stack



OUTPUT

S A D B C

DEPTH 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 |
```

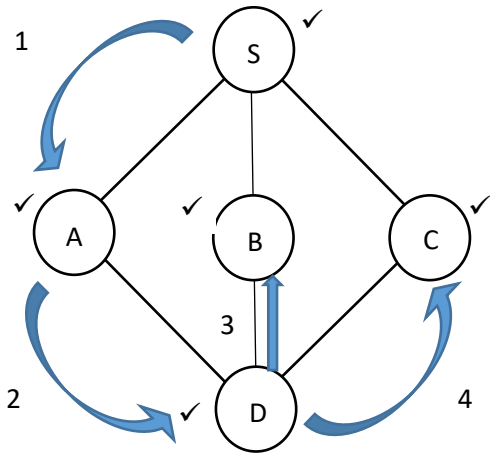
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 1

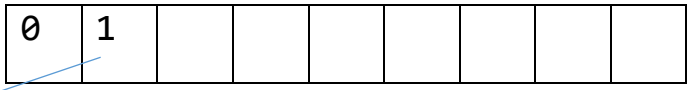
returns -1

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Stack



OUTPUT

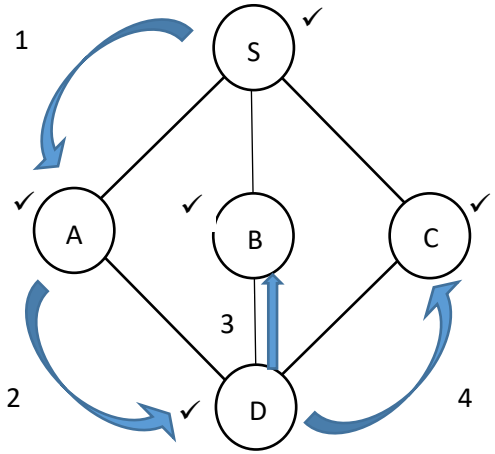
S A D B C

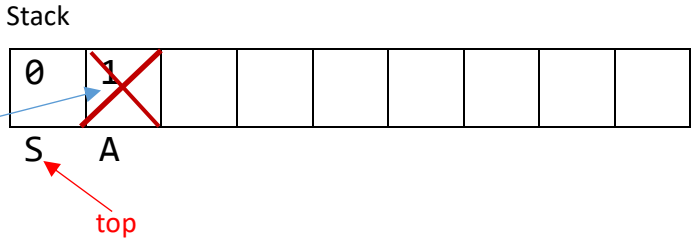
DEPTH FIRST TRAVERSAL

```
                    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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);                          unvisitedVertex=-1
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```





OUTPUT

S A D B C

DEPTH 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 |
```

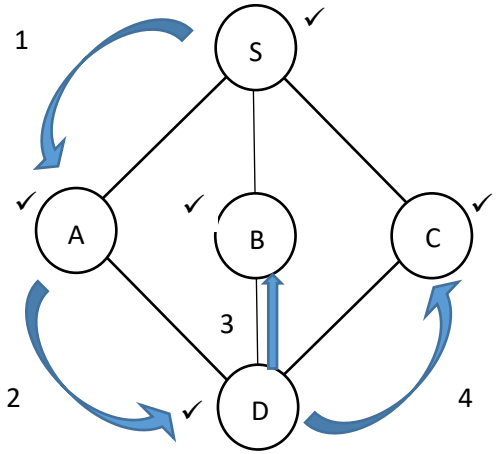
```
int getAdjUnvisitedVertex(int vertexIndex) {
   for (int i = 0; i < vertexCount; i++) {
      if (adjMatrix[vertexIndex][i] == 1 &&
          listVertices[i]->visited == false)
         return i;
   }
   return -1;
}
```

searches in the row no: 0

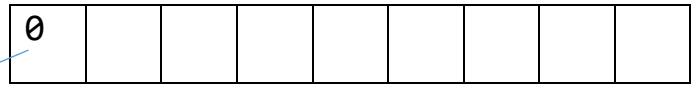
returns -1

```
void depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1



Stack



S

top

OUTPUT

S A D B C

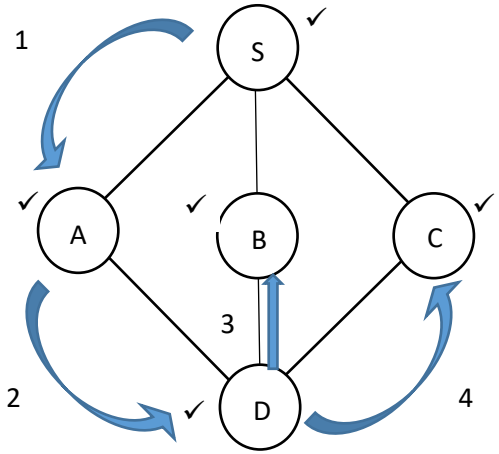
DEPTH FIRST TRAVERSAL

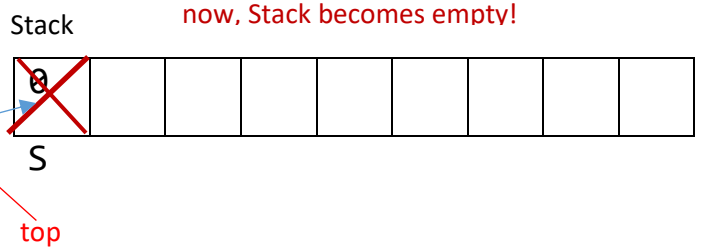
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

unvisitedVertex=-1





OUTPUT

S A D B C

DEPTH FIRST TRAVERSAL

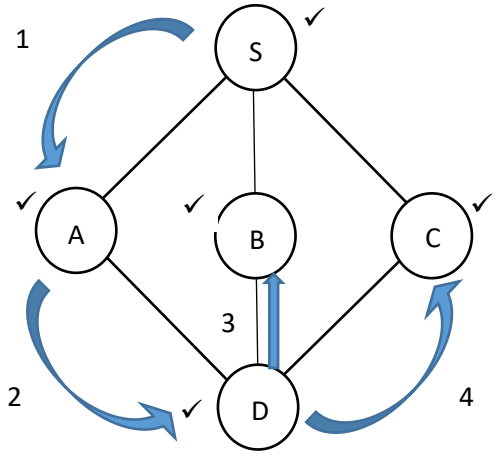
```
                    0   1   2   3   4
listVertices[]  =  {'S','A','B','C','D'}
adjMatrix[][]= 0 S | 0   1   1   1   0 |
               1 A | 1   0   0   0   1 |
               2 B | 1   0   0   0   1 |
               3 C | 1   0   0   0   1 |
               4 D | 0   1   1   1   0 |
```

```
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 depthFirstTraversal(void) {
   int unvisitedVertex;
   stack_t S;
   initializeS(&S);
   lstVertices[0]->visited = true;
   printf("%c ", lstVertices[0]->label);
   push(&S, 0);
   while (!isEmptyS(&S)) {
      unvisitedVertex = getAdjUnvisitedVertex(peek(S));
      if (unvisitedVertex == -1)
         pop(&S);
      else {
         lstVertices[unvisitedVertex]->visited = true;
         printf("%c ", lstVertices[unvisitedVertex]->label);
         push(&S, unvisitedVertex);
      }
   }
}
```

End of function

unvisitedVertex=-1



Stack is empty!

top

OUTPUT

S A D B C