Problem 1: Graph with Nodes S, A, B, C, D

DFS Implementation:

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

#define MAX 5

// Graph represented as an adjacency matrix
int graph[MAX][MAX] = {
    {0, 1, 1, 0, 0}, // S
    {1, 0, 1, 0, 0}, // A
    {1, 1, 0, 0, 1}, // B
    {0, 0, 0, 0, 1}, // C
    {0, 0, 1, 1, 0}  // D
};

int visited[MAX] = {0};

// DFS algorithm
void DFS(int node) {
    visited[node] = 1;
    printf("%c ", node + 'S'); // Convert node number to character

    for (int i = 0; i < MAX; i++) {
        if (graph[node][i] == 1 && !visited[i]) {
            DFS(i);
        }
    }
}

int main() {
    printf("DFS traversal: ");
    DFS(0); // Starting with node S (index 0)
    return 0;
}
```

BFS Implementation:

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

#define MAX 5
```

```
// Graph represented as an adjacency matrix
int graph[MAX][MAX] = {
    {0, 1, 1, 0, 0}, // S
    {1, 0, 1, 0, 0}, // A
    {1, 1, 0, 0, 1}, // B
    {0, 0, 0, 0, 1}, // C
    {0, 0, 1, 1, 0}  // D
};

int visited[MAX] = {0};
int queue[MAX], front = -1, rear = -1;

// BFS algorithm
void BFS(int startNode) {
    front = rear = 0;
    queue[rear] = startNode;
    visited[startNode] = 1;

    while (front <= rear) {
        int node = queue[front++];
        printf("%c ", node + 'S'); // Convert node number to character

        for (int i = 0; i < MAX; i++) {
            if (graph[node][i] == 1 && !visited[i]) {
                queue[++rear] = i;
                visited[i] = 1;
            }
        }
    }
}

int main() {
    printf("BFS traversal: ");
    BFS(0); // Starting with node S (index 0)
    return 0;
}
```

Problem 2:

DFS Implementation:

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

```
struct Node {
    int data;
    struct Node *left, *right;
};

// Function to create a new node
struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

// DFS (Pre-order Traversal)
void DFS(struct Node* root) {
    if (root == NULL)
        return;

    printf("%d ", root->data);
    DFS(root->left);
    DFS(root->right);
}

int main() {
    struct Node* root = newNode(2);
    root->left = newNode(7);
    root->right = newNode(5);
    root->left->left = newNode(2);
    root->left->right = newNode(6);
    root->left->right->left = newNode(10);
    root->left->right->right = newNode(5);
    root->right->right = newNode(9);
    root->right->right->left = newNode(4);
    root->right->right->right = newNode(11);

    printf("DFS traversal (Pre-order): ");
    DFS(root);

    return 0;
}
```

BFS Implementation:

```
  include <stdio.h>
```

```c
#include <stdlib.h>

struct Node {
    int data;
    struct Node *left, *right;
};

// Function to create a new node
struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

// Queue structure for BFS
struct QueueNode {
    struct Node* treeNode;
    struct QueueNode* next;
};

struct Queue {
    struct QueueNode* front;
    struct QueueNode* rear;
};

struct Queue* createQueue() {
    struct Queue* queue = (struct Queue*)malloc(sizeof(struct Queue));
    queue->front = queue->rear = NULL;
    return queue;
}

void enqueue(struct Queue* queue, struct Node* treeNode) {
    struct QueueNode* temp = (struct QueueNode*)malloc(sizeof(struct QueueNode));
    temp->treeNode = treeNode;
    temp->next = NULL;
    if (queue->rear == NULL) {
        queue->front = queue->rear = temp;
        return;
    }
    queue->rear->next = temp;
    queue->rear = temp;
}
```

```
struct Node* dequeue(struct Queue* queue) {
    if (queue->front == NULL)
        return NULL;

    struct QueueNode* temp = queue->front;
    struct Node* treeNode = temp->treeNode;
    queue->front = queue->front->next;
    if (queue->front == NULL)
        queue->rear = NULL;
    free(temp);
    return treeNode;
}

// BFS traversal
void BFS(struct Node* root) {
    if (root == NULL)
        return;

    struct Queue* queue = createQueue();
    enqueue(queue, root);

    while (queue->front != NULL) {
        struct Node* currentNode = dequeue(queue);
        printf("%d ", currentNode->data);

        if (currentNode->left != NULL)
            enqueue(queue, currentNode->left);
        if (currentNode->right != NULL)
            enqueue(queue, currentNode->right);
    }
}

int main() {
    struct Node* root = newNode(2);
    root->left = newNode(7);
    root->right = newNode(5);
    root->left->left = newNode(2);
    root->left->right = newNode(6);
    root->left->right->left = newNode(10);
    root->left->right->right = newNode(5);
    root->right->right = newNode(9);
    root->right->right->left = newNode(4);
    root->right->right->right = newNode(11);
```

```
    printf("BFS traversal: ");
    BFS(root);

    return 0;
}
```