22/02/24

BFS Week-9

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

#define size 7

void push(int a);
int pop();
void display();
void bfs(int graph[][7]);

int fpos = -1, rpos = -1;
int queue[size];

int main(){
    int adj_matrix[7][7] = {
        {0,1,0,1,0,0,0},
        {1,0,1,1,0,1,1},
        {0,1,0,1,1,1,0},
        {1,1,1,0,0,0,0},
        {0,0,1,0,0,0,1},
        {0,1,1,0,0,0,0},
        {0,1,0,0,1,0,0},
    };
    for(int i=0; i<7; i++) queue[i]=0;
    bFS(adj_matrix);
    return 0;
}
```

```
void bfs(int graph[][7]){
    int visited[7];
    for (int i=0; i<7; i++) visited[i]=0;
    push(0); visited[0]=1;
    while (fpos != size){
        for(int i=0;i<7;i++){
            if(graph[queue[fpos]][i]==1 &&
              visited[i]!=1){
                push(i);
                visited[i]=1;
            }}
        }
        printf("%d", pop());
    }
}
void push(int a){
    if (fpos ==-1 && rpos==-1){
        queue[++rpos]=a;
        fpos++;
        return;
    }
    else if(rpos==size-1){
        printf("Queue overflow condition");
        return;
    }
    else
        queue[++rpos]=a;
        return;
    }
```

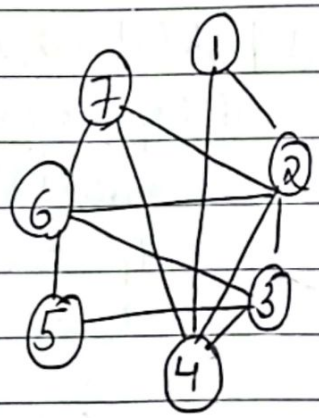
```
int pop(){
    if (fpos == -1){
        printf("Queue underflow condition");
    }
    int n = queue[fpos];
    queue[fpos] = (int) 0;
    fpos++;
    return n;
}

void display(){
    printf("Queue: ");
    for(int i=0; i<size; i++)
        printf("%d", queue[i]);
    printf("\n");
}
```

Output

~~0 1 2 3 4 6 5~~

0 1 3 2 5 6 4



DFS

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

#define size 7

int pos=-1;
int stack[size];

void push(int a);
int pop();
void display();
void dfs(int graph[][7]);

int main(){
    int adj_matrix[7][7]={
        {0,1,0,1,0,0,0},
        {1,0,1,1,0,1,1},
        {0,1,0,1,1,1,0},
        {1,1,1,0,0,0,0},
        {0,0,1,0,0,0,1},
        {0,1,1,0,0,0,0},
        {0,1,0,0,1,0,0},
    };
    for(int i=0;i<7;i++) stack[i]=NULL;
    dfs(adj_matrix);
    return 0;
}
```

```
void dfs (int graph[][7]){
    int visited[7];
    for (i=0; i<7; i++) visited[i] = 0;
    push(0);
    while (pos != -1) {
        bool new_node = false;
        for (i=0; i<7; i++){
            if(graph[stack[pos]][i] == 1 && visited
                new_node = true;
                push(i);
                visited[i] = 1;
                break;
            }
        }
    }
void push (int a) {
    if (pos == size-1){
        printf("Overflow");
        return;
    }
    stack[++pos] = a;
}

int pop() {
    if (pos == -1){
        printf ("Underflow");
        return (int) NULL;
    }
    return stack[pos--];
}
```

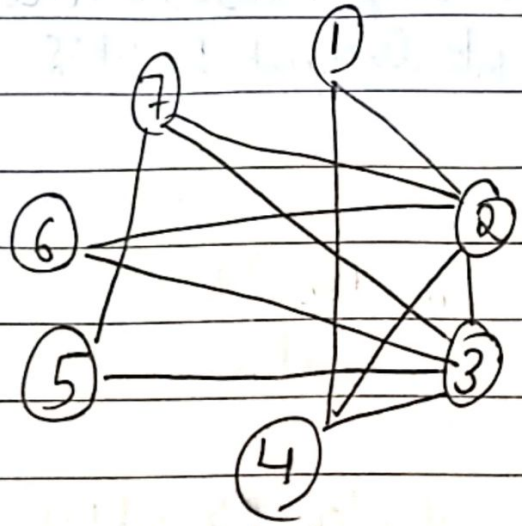
```
void display(){
    for(int i=0; i<size; i++){
        printf("%d", stack[i]);
    }
    printf("\n");
}
```

Output

0 1 2 3 4 6 5

Ex:1

22/2/24.



Swap nodes hackerrank

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

struct node{
    int data;
    struct node *left;
    struct node *right;
};

void swap_nodes_at_level(struct node* root,int
  int level,int height){

    struct node *node;
    if(!root){
        return;
    }
    if(level>height){
        return;
    }

    if(!(level%2)){
        tnode=root->left;
        root->left=root->right;
        root->right=tnode;
    }
```

- sum(p → nodes_at_level (root → left, n, level+1, height);
  sum(p_nodes_at_level (root → right, n, level+1, height);