Zadanie 2

Dodaj do programu dfs.c zmodyfikowanemu w zadaniu 1 poniższą funkcję:

```
void bfs(Graph* g, int a) {
  bool* visited = calloc(g->n, sizeof(bool));
  for (unsigned k = 0; k < g->n; ++k)
      visited[k] = false;
  Queue q; initQueue(&q);
  visited[a] = true;
  displayVertex(a);
  inject(&q, a); // wstaw na końcu
  while (!queueEmpty(q)) {
    int b = front(&q); // pobierz pierwszy wierzchołek
    eject(&q); // usuń go z kolejki
    int c; // dopóki ma nie odwiedzonych sąsiadów
    while ((c = getUnVisitedVertex(g, b, visited)) != -1){
      visited[c] = true;
      displayVertex(c);
      inject(&q, c);
    }
  } // while(kolejka nie jest pusta)
  free(visited);
}
```

Aby funkcja działała, do programu trzeba dodać też kolejkę (Queue). Dwie przykładowe implementacje kolejki znajdują się poniżej:

Implementacja tablicowa:

```
int head=0 , tail =0;
# define MaxElt 200
int Kolejka[ MaxElt +1 ];
void initKolejka () {
        head=0;
        tail=0;
}
void wstaw ( int x ) {
        Kolejka[tail++]=x;
        if ( tail>MaxElt ) tail=0;
}
int obsluz ( int *w) {
        if ( head== tail ) return -1;
        *w = Kolejka[head++];
        if ( head>MaxElt ) head=0;
        return 1;
}
int pusta() {
        if ( head== tail ) return 1;
        else return 0;
}
```

Implementacja dynamiczna:

```
# i f n d e f ITEM_H
# def ine ITEM_H
typedef s t r u c t ITEM {
i n t data ;
s t r u c t ITEM_ next ;
} Item ;
# e n d i f


# i f n d e f QUEUE_H
# def ine QUEUE_H
# inc lude <stdbool . h>
typedef s t r u c t {
Item_ f i r s t ;
Item_ l a s t ;
} Queue ;
# e n d i f

void ini tQueue (Queue _q ) {
q–> f i r s t = q–>l a s t = NULL;
} / / i n i c j a l i z u j e kolej ke
bool queueEmpty (Queue q ) {
r e t u r n ( q . f i r s t == NULL ) ;
} / / zwraca t rue , j e z e l i kolej ka pusta
i n t f r o n t (Queue const _ q ) {
r e t u r n q–> f i r s t –>data ;
} / / pobiera wartosc pierwszego elementu
void clearQueue (Queue_ q ) {
whi le ( q–> f i r s t != NULL) e j e c t ( q ) ;
} / / usuwa z k o l e j k i wszystkie elementy

void i n j e c t (Queue_ q , i n t w)
{
Item_ p = mal loc ( s i ze o f ( Item ) ) ;
p–>data = w;
p–>next = NULL;
i f ( q–> f i r s t == NULL) {
q–> f i r s t = q–>l a s t = p ;
} else {
q–>l a s t –>next = p ;
q–>l a s t = p ;
} / / dodaje element do k o l e j k i
}

void e j e c t (Queue_ q )
{
/ / jeden element lub kolej ka pusta
i f ( q–> f i r s t == q–>l a s t ) {
f ree ( q–> f i r s t ) ;
q–> f i r s t = q–>l a s t = NULL;
} else {
Item_ p = q–> f i r s t ;
q–> f i r s t = p–>next ;
f ree ( p ) ;
}
} / / usuwa element z k o l e j k i
```

Prześlij na moodle zmodyfikowany kod źródłowy programu.