```c
#ifndef SELECTION
#define SELECTION

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

#include "../datatypes/enum.h"

#include "print.h"

char *toLower(char *string)
{
    unsigned char *char_ptr = (unsigned char *)string;

    while (*char_ptr)
    {
        *char_ptr = tolower(*char_ptr);
        char_ptr++;
    }
}

bool ShapeSelection(enum shape *shape)
{
    char *input;

    ShapeSelectionInstructions();

    while (true)
    {
        if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
        {
            NoMemoryAlert();
            exit(1);
        }
        fgets(input, 100 * sizeof(char), stdin);
        toLower(input);

        if (strcmp(input, "rectangle\n") == 0 || strcmp(input, "1\n") == 0)
        {
            free(input);
            *shape = Rectangle;
            return true;
        }
        else if (strcmp(input, "square\n") == 0 || strcmp(input, "2\n") == 0)
        {
            free(input);
            *shape = Square;
            return true;
        }
        else if (strcmp(input, "circle\n") == 0 || strcmp(input, "3\n") == 0)
        {
            free(input);
            *shape = Circle;
            return true;
        }
        else if (strcmp(input, "back\n") == 0)
        {
            free(input);
            return false;
        }
        else if (strcmp(input, "exit\n") == 0)
        {
            free(input);
            exit(0);
        }
        else
        {
            WrongShapeInput();
        }
    }
}

bool ObjectSelection(enum shape *shape)
{
    char *input;

    ObjectSelectionInstructions();

    while (true)
    {
        if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
        {
            NoMemoryAlert();
            exit(1);
        }
        fgets(input, 100 * sizeof(char), stdin);
        toLower(input);

        if (strcmp(input, "cuboid\n") == 0 || strcmp(input, "1\n") == 0)
        {
            free(input);
            *shape = Cuboid;
            return true;
        }
        else if (strcmp(input, "cube\n") == 0 || strcmp(input, "2\n") == 0)
        {
            free(input);
            *shape = Cube;
            return true;
        }
        else if (strcmp(input, "sphere\n") == 0 || strcmp(input, "3\n") == 0)
        {
            free(input);
            *shape = Sphere;
            return true;
        }
        else if (strcmp(input, "cone\n") == 0 || strcmp(input, "4\n") == 0)
        {
            free(input);
            *shape = Cone;
            return true;
        }
        else if (strcmp(input, "back\n") == 0)
        {
            free(input);
            return false;
        }
        else if (strcmp(input, "exit\n") == 0)
        {
            free(input);
            exit(0);
        }
        else
        {
            WrongObjectInput();
        }
    }
}

bool GeometrySelection(enum shape *shape, int dimension)
{
    switch (dimension)
    {
    case 2:
        return ShapeSelection(&(*shape));
        break;
    case 3:
        return ObjectSelection(&(*shape));
        break;
    }
    return false;
}

void DimensionSelection(int *dimension)
{
    char *input;

    DimensionSelectionInstructions();

    while (true)
    {
        if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
        {
            NoMemoryAlert();
            exit(1);
        }
        fgets(input, 100 * sizeof(char), stdin);
        toLower(input);

        if (strcmp(input, "2d\n") == 0 || strcmp(input, "1\n") == 0)
        {
            free(input);
            *dimension = 2;
            return;
        }
        else if (strcmp(input, "3d\n") == 0 || strcmp(input, "2\n") == 0)
        {
            free(input);
            *dimension = 3;
            return;
        }
        else if (strcmp(input, "exit\n") == 0)
        {
            free(input);
            exit(0);
        }
        else
        {
            WrongDimensionInput();
        }
    }
}

void UnitSelection(enum unit *unit)
{
    char *input;

    UnitSelectionInstructions();

    while (true)
    {
        if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
        {
            NoMemoryAlert();
            exit(1);
        }
        fgets(input, 100 * sizeof(char), stdin);
        toLower(input);

        if (strcmp(input, "m\n") == 0 || strcmp(input, "1\n") == 0)
        {
            *unit = m;
            free(input);
            return;
        }
        else if (strcmp(input, "dm\n") == 0 || strcmp(input, "2\n") == 0)
        {
            *unit = dm;
            free(input);
            return;
        }
        else if (strcmp(input, "cm\n") == 0 || strcmp(input, "3\n") == 0)
        {
            *unit = cm;
            free(input);
            return;
        }
        else if (strcmp(input, "mm\n") == 0 || strcmp(input, "4\n") == 0)
        {
            *unit = mm;
            free(input);
            return;
        }
        else
        {
            WrongUnitInput();
        }
    }
}

bool ProcessSelection()
{
    char *input;

    ProcessSelectionInstructions();

    while (true)
    {
        if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
        {
            NoMemoryAlert();
            exit(1);
        }
        fgets(input, 100 * sizeof(char), stdin);
        toLower(input);

        if (strcmp(input, "history\n") == 0 || strcmp(input, "1\n") == 0)
        {
            free(input);
            return true;
        }
        else if (strcmp(input, "calculate\n") == 0 || strcmp(input, "2\n") == 0)
        {
            free(input);
            return false;
        }
        else if (strcmp(input, "exit\n") == 0 || strcmp(input, "3\n") == 0)
        {
            free(input);
            exit(0);
        }
        else
        {
            WrongProcessInput();
        }
    }
}

void ShapeAndObjectSelection(enum shape *shape)
{
    char *input;

    ShapeAndObjectSelectionInstructions();

    while (true)
    {

        if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
        {
            NoMemoryAlert();
            exit(1);
        }
        fgets(input, 100 * sizeof(char), stdin);
        toLower(input);

        if (strcmp(input, "rectangle\n") == 0 || strcmp(input, "1\n") == 0)
        {
            free(input);
            *shape = Rectangle;
            return;
        }
        else if (strcmp(input, "square\n") == 0 || strcmp(input, "2\n") == 0)
        {
            free(input);
            *shape = Square;
            return;
        }
        else if (strcmp(input, "circle\n") == 0 || strcmp(input, "3\n") == 0)
        {
            free(input);
            *shape = Circle;
            return;
        }
        else if (strcmp(input, "cuboid\n") == 0 || strcmp(input, "4\n") == 0)
        {
            free(input);
            *shape = Cuboid;
            return;
        }
        else if (strcmp(input, "cube\n") == 0 || strcmp(input, "5\n") == 0)
        {
            free(input);
            *shape = Cube;
            return;
        }
        else if (strcmp(input, "sphere\n") == 0 || strcmp(input, "6\n") == 0)
        {
            free(input);
            *shape = Sphere;
            return;
        }
        else if (strcmp(input, "cone\n") == 0 || strcmp(input, "7\n") == 0)
        {
            free(input);
            *shape = Cone;
            return;
        }
        else if (strcmp(input, "exit\n") == 0)
        {
            free(input);
            exit(0);
        }
        else
        {
            WrongShapeAndObjectInput();
        }
    }
}

#endif
```