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

typedef struct stackNode {   /* self-referential structure */
    int data;
    struct stackNode* nextPtr;
} StackNode_t;

void push(StackNode_t**, int);
int pop(StackNode_t**);
int isEmpty(StackNode_t*);
void printStack(StackNode_t*);
void instructions(void);

int main()
{
    StackNode_t* stackPtr = NULL;  /* points to stack top */
    int choice, value;

    instructions();
    printf("? ");
    scanf("%d", &choice);

    while (choice != 3) {

        switch (choice) {
        case 1:      /* push value onto stack */
            printf("Enter an integer: ");
            scanf("%d", &value);
            push(&stackPtr, value);
            printStack(stackPtr);
            break;
        case 2:      /* pop value off stack */
            if (!isEmpty(stackPtr))
                printf("The popped value is %d.\n",
                    pop(&stackPtr));

            printStack(stackPtr);
            break;
        default:
            printf("Invalid choice.\n\n");
            instructions();
            break;
        }

        printf("? ");
        scanf("%d", &choice);
    }

    printf("End of run.\n");
    return 0;
}

```

```c
/* Print the instructions */
void instructions(void)
{
    printf("Enter choice:\n"
        "1 to push a value on the stack\n"
        "2 to pop a value off the stack\n"
        "3 to end program\n");
}

/* Insert a node at the stack top */
void push(StackNode_t** topPtr, int info)
{
    StackNode_t* newPtr;

    newPtr = malloc(sizeof(StackNode_t));
    if (newPtr != NULL) {
        newPtr->data = info;
        newPtr->nextPtr = *topPtr;
        *topPtr = newPtr;
    }
    else
        printf("%d not inserted. No memory available.\n",
            info);
}

/* Remove a node from the stack top */
int pop(StackNode_t** topPtr)
{
    StackNode_t* tempPtr;
    int popValue;

    tempPtr = *topPtr;
    popValue = (*topPtr)->data;
    *topPtr = (*topPtr)->nextPtr;
    free(tempPtr);
    return popValue;
}

/* Print the stack */
void printStack(StackNode_t* currentPtr)
{
    if (currentPtr == NULL)
        printf("The stack is empty.\n\n");
    else {
        printf("The stack is:\n");

        while (currentPtr != NULL) {
            printf("%d --> ", currentPtr->data);
            currentPtr = currentPtr->nextPtr;
        }

        printf("NULL\n\n");
    }
}

/* Is the stack empty? */
int isEmpty(StackNode_t* topPtr)
{
    return topPtr == NULL;
}

```