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

#define DAYS 3

struct activity {
    char *nday;  // Day name (e.g., Monday)
    int dday;    // Date of the day
    char *desc;  // Activity description
};

typedef struct activity Plan;

Plan* create() {
    Plan *t = (Plan *) malloc(sizeof(Plan) * DAYS);  // Allocate for DAYS (3)
    if (t == NULL) {
        printf("Sufficient memory not allocated\n");
        return NULL;  // Return NULL if memory allocation fails
    }
    return t;
}

void read(Plan *p) {
    int i;
    for (i = 0; i < DAYS; i++) {
        p[i].nday = (char *) malloc(9 * sizeof(char));  // Allocate memory for day name
        if (p[i].nday == NULL) {
            printf("Memory allocation failed for day name.\n");
            return;  // Return if memory allocation fails
        }

        printf("Enter name of the day: ");
        scanf(" %s", p[i].nday);  // Read day name

        printf("Enter date of the day: ");
        scanf("%d", &(p[i].dday));  // Read date of the day

        printf("Enter description of the activity: ");
        p[i].desc = (char *) malloc(400 * sizeof(char));  // Allocate memory for description
        if (p[i].desc == NULL) {
            printf("Memory allocation failed for description.\n");
            return;  // Return if memory allocation fails
        }

        // Read activity description
        scanf(" %[^\n]", p[i].desc);

        // Reallocate to reduce the memory used for the description (trim excess)
        p[i].desc = (char *) realloc(p[i].desc, strlen(p[i].desc) + 1);
        if (p[i].desc == NULL) {
            printf("Memory reallocation failed for description.\n");
            return;  // Return if reallocation fails
        }
    }
}

void display(Plan *p) {
    int i;
    printf("**** Activity description for %d days ****\n", DAYS);
    for (i = 0; i < DAYS; i++) {
        printf("\nName of the day: %s", p[i].nday);
        printf("\nDate of the day: %d", p[i].dday);
        printf("\nActivity description: %s\n", p[i].desc);
    }
}

int main() {
    Plan* cal = NULL;
    cal = create();

    if (cal == NULL) {
        printf("Memory allocation failed.\n");
        return -1;  // Exit the program if memory allocation fails
    }

    read(cal);
    display(cal);
    return 0;
}
```