/*1. Develop a Program in C for the following:
a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).
b) Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.*/

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_DAYS 7
struct Day
{
   char* dayName;
   int date;
   char* activityDescription;
};

void create(struct Day* calendar )
 {
   for (int i = 0; i < MAX_DAYS; i++)
    {
      calendar[i].dayName = (char *)malloc(20 * sizeof(char)); // Allocate memory for name
      calendar[i].activityDescription = (char *)malloc(100 * sizeof(char)); //alocate memory
                                                                          //for description
   }
}
void read(struct Day* calendar)
{
     for (int i = 0; i < MAX_DAYS; ++i)
     {
          printf("\nEnter details for day %d:\n", i + 1);
          printf("Enter day name: ");
          scanf("%s", calendar[i].dayName);
          printf("Enter date: ");
          scanf("%d", &calendar[i].date);
```

```
        if (calendar[i].date<=0 || calendar[i].date>31)
        {
                printf("Please enter Valid Date! (Any integer from 1 to 31)\n");
                exit(0);
         }
        __fpurge(stdin);
        printf("Enter activity description: ");
        scanf(" %[^\n]", calendar[i].activityDescription);
    }
}
// Function to display the calendar
void display(struct Day* calendar)
{
        printf("\nWeek's Activity Details Report:\n");
        printf("%-10s %-10s %-30s\n", "Day", "Date", "Activity Description");
        for(int i = 0; i < MAX_DAYS; ++i)
        {
             printf("%-10s %-10d % 30s\n", calendar[i].dayName, calendar[i].date,
           calendar[i].activityDescription);
        }
}
//Function to deallocate the memory
void freeMemory(struct Day* calendar)
{
        for (int i = 0; i < MAX_DAYS; ++i)
        {
                free(calendar[i].dayName);
                free(calendar[i].activityDescription);
        }
}
```

```
int main()
{
   // Dynamically allocate memory for the calendar
     struct Day*calendar=(struct Day*)malloc(sizeof(struct Day)*MAX_DAYS);

     if (calendar == NULL)
     {
              printf("Memory allocation failed.\n");
              return 1;
     }
   create(calendar);
   read(calendar);
   display(calendar);
   freeMemory(calendar);
   free(calendar);
   return 0;
}
```

Output