**Program 1**

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 a 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.**

Program:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Structure to represent a day

struct Day {

char \*name;

int date;

char \*activity;

};

// Function to create a day

struct Day createDay() {

struct Day day;

// Allocate memory for day name

day.name = (char \*)malloc(sizeof(char) \* 20);

printf("Enter day name: ");

scanf("%s", day.name);

printf("Enter date: ");

scanf("%d", &day.date);

// Allocate memory for activity description

day.activity = (char \*)malloc(sizeof(char) \* 100);

printf("Enter activity: ");

scanf(" %[^\n]s", day.activity);

return day;

}

// Function to read the calendar data from the keyboard

void readCalendar(struct Day \*calendar, int size) {

for (int i = 0; i < size; ++i) {

calendar[i] = createDay();

}

}

// Function to display the calendar

void displayCalendar(struct Day \*calendar, int size) {

for (int i = 0; i < size; ++i) {

printf("Day: %s\n", calendar[i].name);

printf("Date: %d\n", calendar[i].date);

printf("Activity: %s\n\n", calendar[i].activity);

}

}

// Function to free memory allocated for the calendar

void freeCalendar(struct Day \*calendar, int size) {

for (int i = 0; i < size; ++i) {

free(calendar[i].name);

free(calendar[i].activity);

}

}

int main() {

int size;

// Get the number of days in a week

printf("Enter the number of days in a week: ");

scanf("%d", &size);

// Dynamically allocate memory for the calendar

struct Day \*calendar = (struct Day \*)malloc(sizeof(struct Day) \* size);

// Check if memory allocation is successful

if (calendar == NULL) {

printf("Memory allocation failed.\n");

return 1; // Exit the program with an error code

}

// Read calendar data from the keyboard

readCalendar(calendar, size);

// Display calendar details

printf("\nCalendar Details:\n");

displayCalendar(calendar, size);

// Free memory allocated for the calendar

freeCalendar(calendar, size);

free(calendar);

return 0; // Exit the program successfully

}