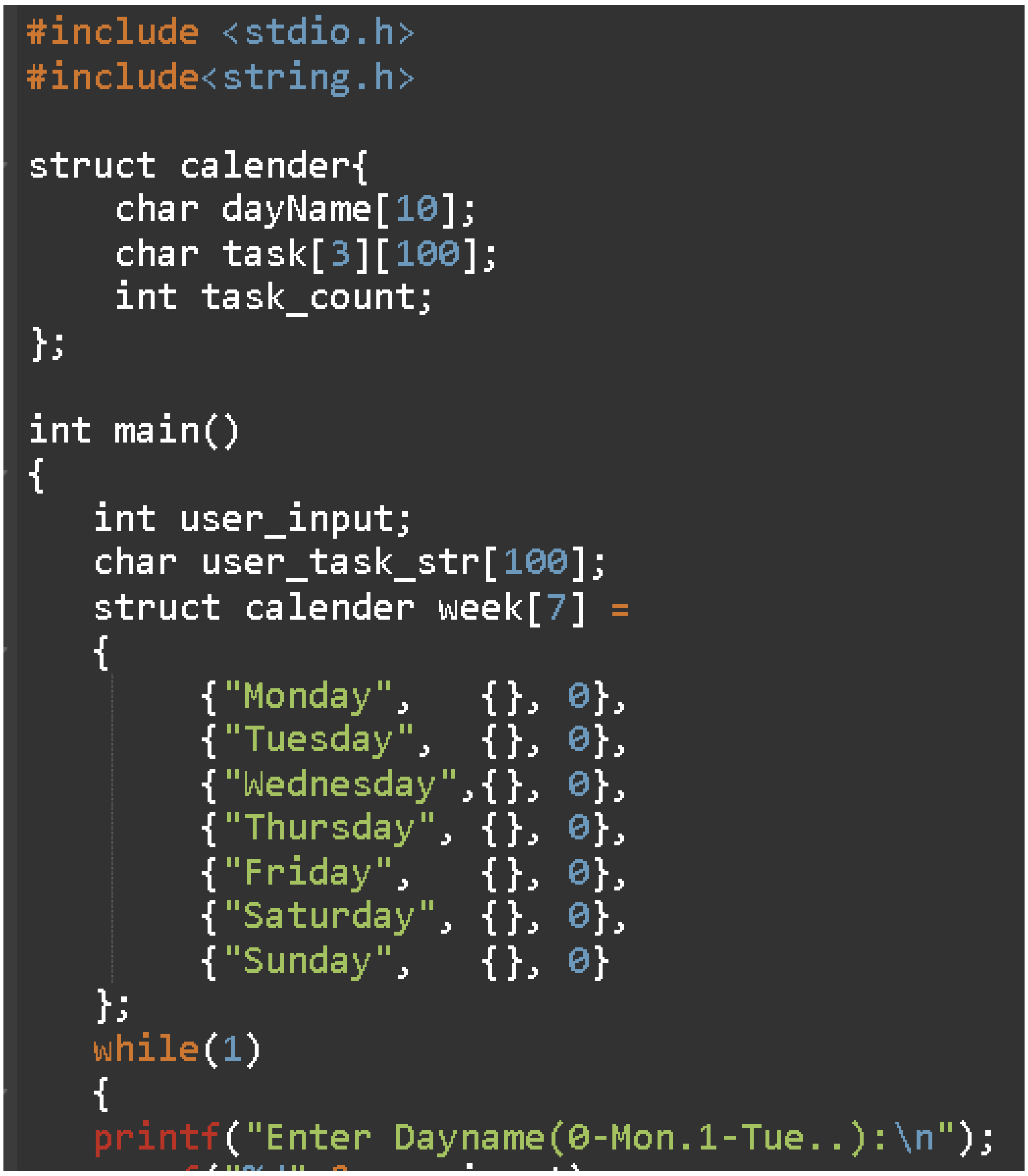
Advanced C Assignment 1

Sri Gnana Saravan.N

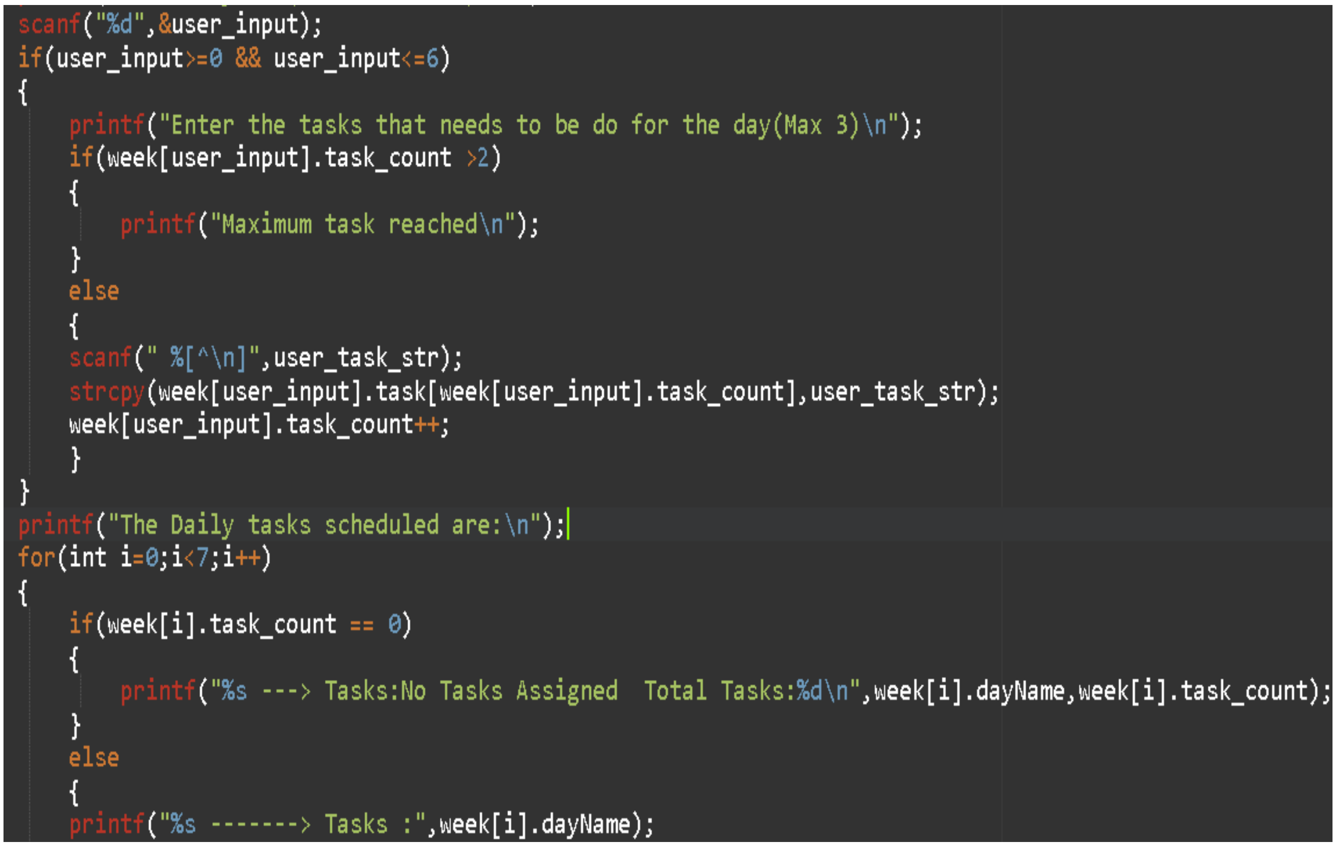
VIT Chennai

1. Write a C program that represents a calendar for a week. Each day has: 1.dayName (e.g., 'Monday') 2.tasks (array of strings with maximum 3 tasks per day) Note: Define appropriate structures. Allow the user to input tasks for any day. Display all tasks grouped by the day.?

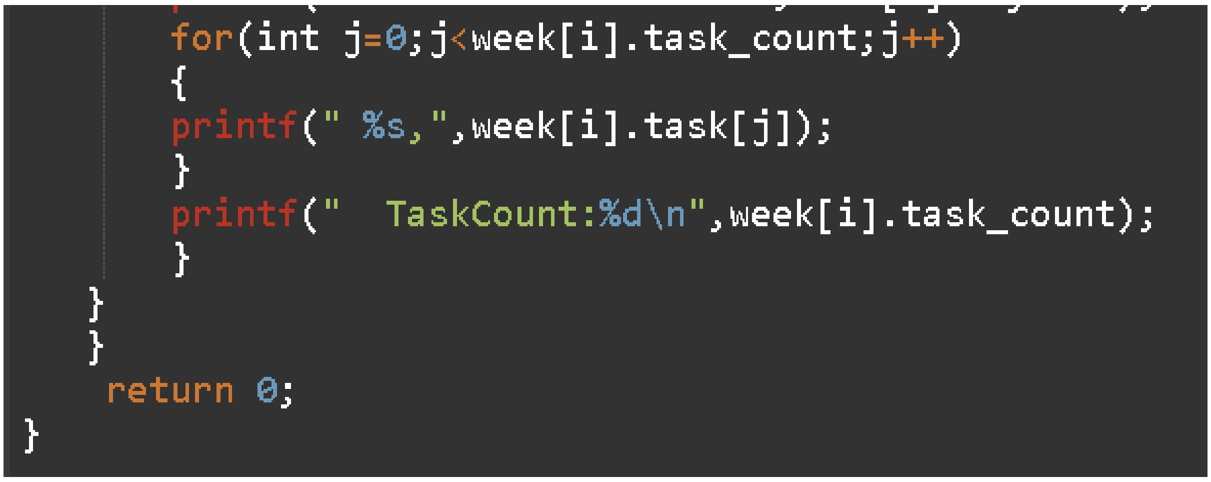
Solution:



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

struct calender{
    char dayName[10];
    char task[3][100];
    int task_count;
};

int main()
{
   int user_input;
   char user_task_str[100];
   struct calender week[7] =
   {
        {"Monday",   {}, 0},
        {"Tuesday",  {}, 0},
        {"Wednesday",{}, 0},
        {"Thursday", {}, 0},
        {"Friday",   {}, 0},
        {"Saturday", {}, 0},
        {"Sunday",   {}, 0}
   };
   while(1)
   {
   printf("Enter Dayname(0-Mon.1-Tue..):\n");
```

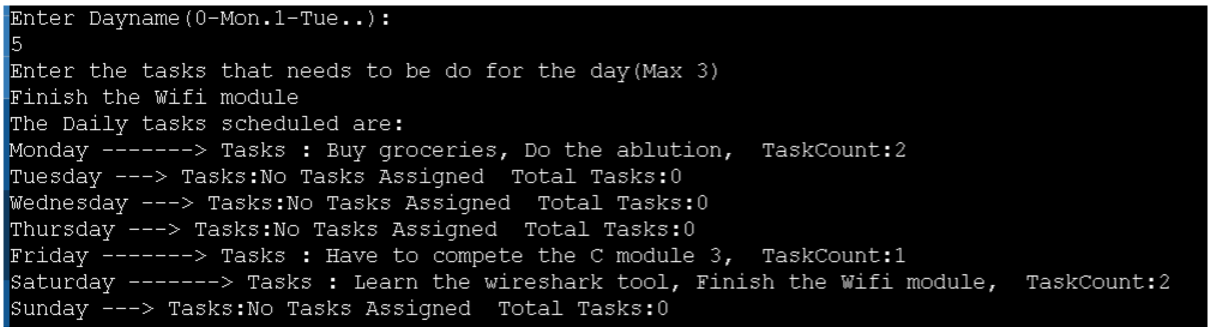


```c
scanf("%d",&user_input);
if(user_input>=0 && user_input<=6)
{
    printf("Enter the tasks that needs to be do for the day(Max 3)\n");
    if(week[user_input].task_count >2)
    {
        printf("Maximum task reached\n");
    }
    else
    {
    scanf(" %[^\n]",user_task_str);
    strcpy(week[user_input].task[week[user_input].task_count],user_task_str);
    week[user_input].task_count++;
    }
}
printf("The Daily tasks scheduled are:\n");
for(int i=0;i<7;i++)
{
    if(week[i].task_count == 0)
    {
        printf("%s ---> Tasks:No Tasks Assigned  Total Tasks:%d\n",week[i].dayName,week[i].task_count);
    }
    else
    {
    printf("%s -------> Tasks :",week[i].dayName);
```

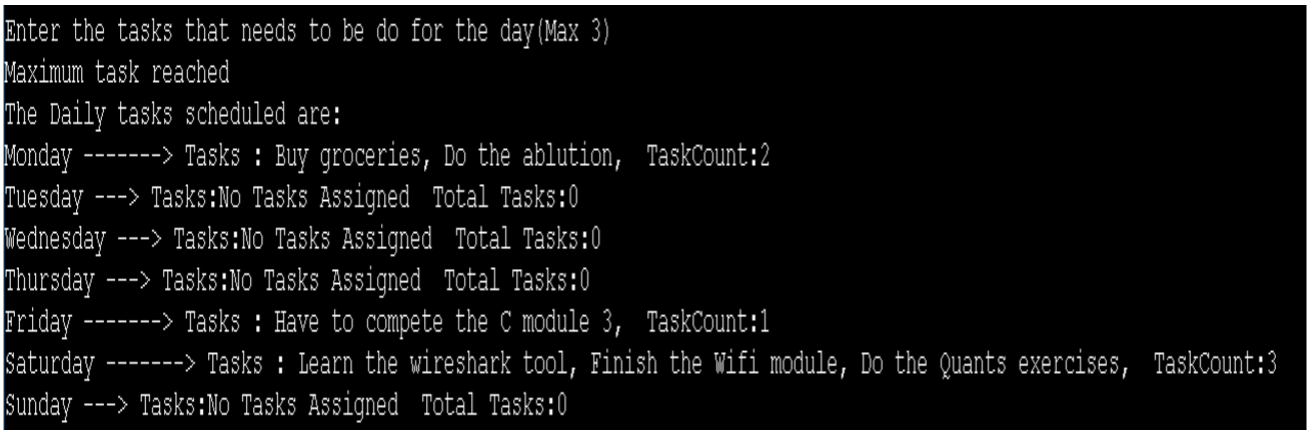


```c
        for(int j=0;j<week[i].task_count;j++)
        {
        printf(" %s,",week[i].task[j]);
        }
        printf("  TaskCount:%d\n",week[i].task_count);
        }
    }
    }
     return 0;
}
```

Output:



```
Enter Dayname(0-Mon.1-Tue..):
5
Enter the tasks that needs to be do for the day(Max 3)
Finish the Wifi module
The Daily tasks scheduled are:
Monday -------> Tasks : Buy groceries, Do the ablution,  TaskCount:2
Tuesday ---> Tasks:No Tasks Assigned  Total Tasks:0
Wednesday ---> Tasks:No Tasks Assigned  Total Tasks:0
Thursday ---> Tasks:No Tasks Assigned  Total Tasks:0
Friday -------> Tasks : Have to compete the C module 3,  TaskCount:1
Saturday -------> Tasks : Learn the wireshark tool, Finish the Wifi module,  TaskCount:2
Sunday ---> Tasks:No Tasks Assigned  Total Tasks:0
```



```
Enter the tasks that needs to be do for the day(Max 3)
Maximum task reached
The Daily tasks scheduled are:
Monday -------> Tasks : Buy groceries, Do the ablution,  TaskCount:2
Tuesday ---> Tasks:No Tasks Assigned  Total Tasks:0
Wednesday ---> Tasks:No Tasks Assigned  Total Tasks:0
Thursday ---> Tasks:No Tasks Assigned  Total Tasks:0
Friday -------> Tasks : Have to compete the C module 3,  TaskCount:1
Saturday -------> Tasks : Learn the wireshark tool, Finish the Wifi module, Do the Quants exercises,  TaskCount:3
Sunday ---> Tasks:No Tasks Assigned  Total Tasks:0
```

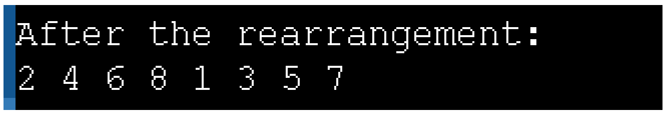
2. Write a function in C that takes a pointer to an integer array and its size, and then rearra nges the array in-place such that all even numbers appear before odd numbers, preserving the original relative order using only pointer a rithmetic (no indexing with []).

Solution:

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

int main()
{
   int n;
   printf("Enter the array size:\n");
   scanf("%d",&n);
   int array[n];
   for(int i=0;i<n;i++)
   {
      printf("Enter the element:\n");
      scanf("%d",array+i);
   }

   int* end = array + n;
```

```
    for (int* curr = array; curr < end; curr++) {
        if (*curr % 2 == 0) {
            int* scan = curr;
            while (scan > array && (*(scan - 1) % 2 != 0)) {
                int temp = *scan;
                *scan = *(scan - 1);
                *(scan - 1) = temp;
                scan--;
            }
        }
}
 printf("After the rearrangement:\n");
  for(int i=0;i<n;i++)
    {
        printf("%d ",*(array+i));
    }
return 0;
}
```

Output:



```
After the rearrangement:
2 4 6 8 1 3 5 7
```

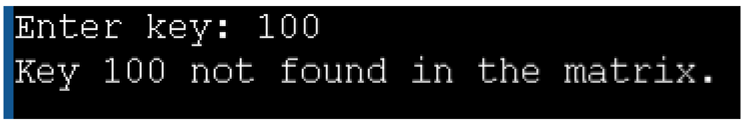
3. You are given a 2D matrix of size nx n where each row and each column is sorted in increasing order. Write a C function to determine whether a given key exists in the matrix using the most efficient approach.?

Solution:

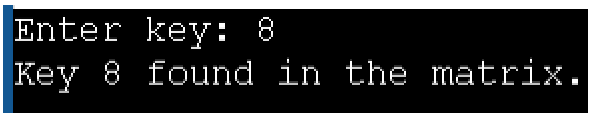
```
#include <stdio.h>

#include <stdbool.h>
```

```
bool searchSortedMatrix(int* matrix, int n, int key) {
    int row = 0;
    int col = n - 1;

    while (row < n && col >= 0) {
        int current = *(matrix + row * n + col);
        if (current == key) {
            return true;
        } else if (current > key) {
            col--;
        } else {
            row++;
        }
    }

    return false;
}


int main() {
    int n = 4;
    int matrix[4][4] = {
        {1,  4,  7, 11},
        {2,  5,  8, 12},
        {3,  6,  9, 16},
        {10,13,14,17}
    };
    int key;
    printf("Enter key: ");
    scanf("%d", &key);

    if (searchSortedMatrix((int*)matrix, n, key)) {
        printf("Key %d found in the matrix.\n", key);
    } else {
        printf("Key %d not found in the matrix.\n", key);
    }

    return 0;
}
```

Output:



```
Enter key: 100
Key 100 not found in the matrix.
```



```
Enter key: 8
Key 8 found in the matrix.
```