Task 1:

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

int main() {
    int *steps, hours, extra, i, threshold;
    int total = 0, max;

    printf("Enter hours: ");
    scanf("%d", &hours);

    steps = malloc(hours * sizeof(int));

    for (i = 0; i < hours; i++) {
        scanf("%d", steps + i);
    }

    printf("Add extra hours: ");
    scanf("%d", &extra);

    if (extra > 0) {
        steps = realloc(steps, (hours + extra) * sizeof(int));
        for (i = hours; i < hours + extra; i++) {
            scanf("%d", steps + i);
        }
        hours += extra;
    }

    max = *(steps + 0);

    for (i = 0; i < hours; i++) {
```

```
            scanf("%d", steps + i);
        }
        hours += extra;
    }

    max = *(steps + 0);

    for (i = 0; i < hours; i++) {
        int v = *(steps + i);
        total += v;
        if (v > max) max = v;
    }

    printf("Enter threshold: ");
    scanf("%d", &threshold);

    int count = 0;
    for (i = 0; i < hours; i++) {
        if (*(steps + i) > threshold) count++;
    }

    FILE *f = fopen("fitness_tracker.txt", "w");
    fprintf(f, "Hours: %d\nTotal: %d\nMax: %d\nAbove %d: %d\n",
            hours, total, max, threshold, count);
    fclose(f);

    printf("Data saved.\n");

    free(steps);
    return 0;
}

```

Task 2:

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

typedef struct {
    char studentName[50];
    char rollNumber[20];
    int seatNumber;
} Student;

int isAlphanumericRecursive(char *str, int index) {
    if (str[index] == '\0')
        return 1; // end -> valid
    if (!isalnum(str[index]))
        return 0; // not alphanumeric
    return isAlphanumericRecursive(str, index + 1);
}

int main() {
    int n, extra, i;
    Student *list;

    printf("Enter number of students: ");
    scanf("%d", &n);

    list = malloc(n * sizeof(Student));

    for (i = 0; i < n; i++) {
        printf("\nStudent %d Name: ", i + 1);
        scanf("%s", list[i].studentName);

```

```
28
29      for (i = 0; i < n; i++) {
30          printf("\nStudent %d Name: ", i + 1);
31          scanf("%s", list[i].studentName);
32
33          do {
34              printf("Roll Number: ");
35              scanf("%s", list[i].rollNumber);
36          } while (!isAlphanumericRecursive(list[i].rollNumber, 0));
37
38          printf("Seat Number: ");
39          scanf("%d", &list[i].seatNumber);
40      }
41
42      printf("\nAdd more students: ");
43      scanf("%d", &extra);
44
45      if (extra > 0) {
46          list = realloc(list, (n + extra) * sizeof(Student));
47
48          for (i = n; i < n + extra; i++) {
49              printf("\nStudent %d Name: ", i + 1);
50              scanf("%s", list[i].studentName);
51
52              do {
53                  printf("Roll Number: ");
54                  scanf("%s", list[i].rollNumber);
55              } while (!isAlphanumericRecursive(list[i].rollNumber, 0));
56
57              printf("Seat Number: ");
58              scanf("%d", &list[i].seatNumber);
59          }
```

```c
    if (extra > 0) {
        list = realloc(list, (n + extra) * sizeof(Student));

        for (i = n; i < n + extra; i++) {
            printf("\nStudent %d Name: ", i + 1);
            scanf("%s", list[i].studentName);

            do {
                printf("Roll Number: ");
                scanf("%s", list[i].rollNumber);
            } while (!isAlphanumericRecursive(list[i].rollNumber, 0));

            printf("Seat Number: ");
            scanf("%d", &list[i].seatNumber);
        }

        n += extra;
    }

    FILE *f = fopen("seating.txt", "w");
    for (i = 0; i < n; i++) {
        fprintf(f, "Name: %s | Roll: %s | Seat: %d\n",
                list[i].studentName, list[i].rollNumber, list[i].seatNumber);
    }
    fclose(f);

    printf("\nSeating plan saved to seating.txt\n");

    free(list);
    return 0;
}

```

Output 2:

```
Enter number of students: 2

Student 1 Name: Shoaib
Roll Number: 2019
Seat Number: 19

Student 2 Name: Hamza
Roll Number: 4404
Seat Number: 04

Add more students: 0

Seating plan saved to seating.txt

--------------------------------
Process exited after 56.27 seconds with return value 0
```

Task 3:

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

int main() {
    float *temp;
    int n, extra, i, alertCount = 0;
    float max, min, threshold;

    printf("Enter number of readings: ");
    scanf("%d", &n);

    temp = malloc(n * sizeof(float));

    printf("Enter temperatures:\n");
    for (i = 0; i < n; i++)
        scanf("%f", temp + i);

    printf("Add more readings: ");
    scanf("%d", &extra);

    if (extra > 0) {
        temp = realloc(temp, (n + extra) * sizeof(float));
        printf("Enter new temperatures:\n");
        for (i = n; i < n + extra; i++)
            scanf("%f", temp + i);
        n += extra;
    }

    max = min = temp[0];

    for (i = 0; i < n; i++) {
        float t = *(temp + i);
```

```c
            scanf("%f", temp + i);
        n += extra;
    }

    max = min = temp[0];

    for (i = 0; i < n; i++) {
        float t = *(temp + i);
        if (t > max) max = t;
        if (t < min) min = t;
    }

    printf("Enter alert threshold: ");
    scanf("%f", &threshold);

    for (i = 0; i < n; i++)
        if (*(temp + i) > threshold)
            alertCount++;

    FILE *f = fopen("temperature_summary.txt", "w");
    fprintf(f, "Total Readings: %d\n", n);
    fprintf(f, "Highest Temperature: %.2f\n", max);
    fprintf(f, "Lowest Temperature: %.2f\n", min);
    fprintf(f, "Readings Above %.2f: %d\n", threshold, alertCount);
    fclose(f);

    printf("\nSummary saved to temperature_summary.txt\n");

    free(temp);
    return 0;
}
```

Output 3:

```
Enter number of readings: 3
Enter temperatures:
36
26
16
Add more readings: 0
Enter alert threshold: 25

Summary saved to temperature_summary.txt

--------------------------------
Process exited after 32.99 seconds with return value 0
Press any key to continue . . .
```

Task 4:

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

typedef struct {
    char bookTitle[100];
    char memberID[30];
    int checkoutTime;       // day number (1-31)
    int returnDueDays;      // days after checkout
    int dueDate;            // computed automatically
} Log;

int main() {
    Log *logs;
    int n, extra, i;

    printf("Enter number of checkout logs: ");
    scanf("%d", &n);

    logs = malloc(n * sizeof(Log));

    for (i = 0; i < n; i++) {
        printf("\nBook Title: ");
        scanf("%s", logs[i].bookTitle);

        printf("Member ID: ");
        scanf("%s", logs[i].memberID);

        printf("Checkout Time (day of month): ");
        scanf("%d", &logs[i].checkoutTime);

        printf("Return Due Days: ");
```

```c

    printf("\nAdd more logs: ");
    scanf("%d", &extra);

    if (extra > 0) {
        logs = realloc(logs, (n + extra) * sizeof(Log));

        for (i = n; i < n + extra; i++) {
            printf("\nBook Title: ");
            scanf("%s", logs[i].bookTitle);

            printf("Member ID: ");
            scanf("%s", logs[i].memberID);

            printf("Checkout Time (day of month): ");
            scanf("%d", &logs[i].checkoutTime);

            printf("Return Due Days: ");
            scanf("%d", &logs[i].returnDueDays);

            logs[i].dueDate = logs[i].checkoutTime + logs[i].returnDueDays;
        }

        n += extra;
    }

    // Append to CSV file
    FILE *f = fopen("library_log.csv", "a");

    if (ftell(f) == 0) {
        // Write header only if file is empty
        fprintf(f, "BookTitle,MemberID,CheckoutDay,ReturnDueDays,DueDate\n");
```

```
55              scanf("%d", &logs[i].returnDueDays);
56
57              logs[i].dueDate = logs[i].checkoutTime + logs[i].returnDueDays;
58          }
59
60          n += extra;
61      }
62
63      // Append to CSV file
64      FILE *f = fopen("library_log.csv", "a");
65
66      if (ftell(f) == 0) {
67          // Write header only if file is empty
68          fprintf(f, "BookTitle,MemberID,CheckoutDay,ReturnDueDays,DueDate\n");
69      }
70
71      for (i = 0; i < n; i++) {
72          fprintf(f, "%s,%s,%d,%d,%d\n",
73                  logs[i].bookTitle,
74                  logs[i].memberID,
75                  logs[i].checkoutTime,
76                  logs[i].returnDueDays,
77                  logs[i].dueDate);
78      }
79
80      fclose(f);
81      free(logs);
82
83      printf("\nLogs saved to library_log.csv\n");
84      return 0;
85  }
```

Output 4:

```
Enter number of checkout logs: 2

Book Title: 1010
Member ID: 2019
Checkout Time (day of month): 5
Return Due Days: 10

Book Title: 0101
Member ID: 9102
Checkout Time (day of month): 2
Return Due Days: 8

Add more logs: 0

Logs saved to library_log.csv

--------------------------------
Process exited after 60.37 seconds with return value 0
Press any key to continue . . .
```

Task 5:

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

int main() {
    int *duration;      // stores rental duration for each customer
    int n, extra, i;
    int rate;           // cost per hour/day
    int total = 0, max;

    printf("Enter cost rate per duration unit (e.g., per hour/day): ");
    scanf("%d", &rate);

    printf("Enter number of customers today: ");
    scanf("%d", &n);

    duration = malloc(n * sizeof(int));

    printf("Enter rental durations:\n");
    for (i = 0; i < n; i++)
        scanf("%d", duration + i);

    printf("Add more customers: ");
    scanf("%d", &extra);

    if (extra > 0) {
        duration = realloc(duration, (n + extra) * sizeof(int));
        printf("Enter new durations:\n");
        for (i = n; i < n + extra; i++)
            scanf("%d", duration + i);
        n += extra;
    }

```

```
33      max = *(duration + 0);
34      for (i = 0; i < n; i++) {
35          int d = *(duration + i);
36          total += d;
37          if (d > max)
38              max = d;
39      }
40
41      FILE *f = fopen("Rental_Invoices.txt", "w");
42
43      fprintf(f, "Rental Duration & Cost Summary\n");
44      fprintf(f, "----------------------------------\n");
45      fprintf(f, "Total Customers: %d\n", n);
46      fprintf(f, "Total Rental Time Today: %d\n", total);
47      fprintf(f, "Highest Rental Duration: %d\n\n", max);
48
49      fprintf(f, "Customer Invoices:\n");
50      fprintf(f, "Duration,Cost\n");
51
52      for (i = 0; i < n; i++) {
53          int cost = (*(duration + i)) * rate;
54          fprintf(f, "%d,%d\n", *(duration + i), cost);
55      }
56
57      fclose(f);
58
59      printf("\nInvoices saved to Rental_Invoices.txt\n");
60
61      free(duration);
62      return 0;
63  }
```

Output 5:

```
Enter cost rate per duration unit (e.g., per hour/day): 4
Enter number of customers today: 2
Enter rental durations:
2
1
Add more customers: 0

Invoices saved to Rental_Invoices.txt

--------------------------------
Process exited after 32.59 seconds with return value 0
```

Task 6:

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


int totalRecursive(int *arr, int index, int n) {
    if (index == n)
        return 0;
    return arr[index] + totalRecursive(arr, index + 1, n);
}

int main() {
    int *gates;
    int n, extra, i;

    printf("Enter number of gates: ");
    scanf("%d", &n);

    gates = malloc(n * sizeof(int));

    printf("Enter headcounts for each gate:\n");
    for (i = 0; i < n; i++)
        scanf("%d", gates + i);

    printf("Add more gates: ");
    scanf("%d", &extra);

    if (extra > 0) {
        gates = realloc(gates, (n + extra) * sizeof(int));
        printf("Enter headcounts for new gates:\n");
        for (i = n; i < n + extra; i++)
            scanf("%d", gates + i);
```

```
    }

    // Find highest attendance gate
    int max = *(gates + 0);
    int maxGate = 1;

    for (i = 0; i < n; i++) {
        if (*(gates + i) > max) {
            max = *(gates + i);
            maxGate = i + 1;
        }
    }

    // Recursive total calculation
    int total = totalRecursive(gates, 0, n);

    // Save to file with timestamp
    FILE *f = fopen("attendance.txt", "a");

    time_t now = time(NULL);
    fprintf(f, "\n--- Attendance Log (%s) ---\n", ctime(&now));

    fprintf(f, "Gate Data:\n");
    for (i = 0; i < n; i++)
        fprintf(f, "Gate %d: %d attendees\n", i + 1, *(gates + i));

    fprintf(f, "Total Attendees: %d\n", total);
    fprintf(f, "Gate with Highest Attendance: Gate %d (%d attendees)\n",
            maxGate, max);

    fclose(f);

```

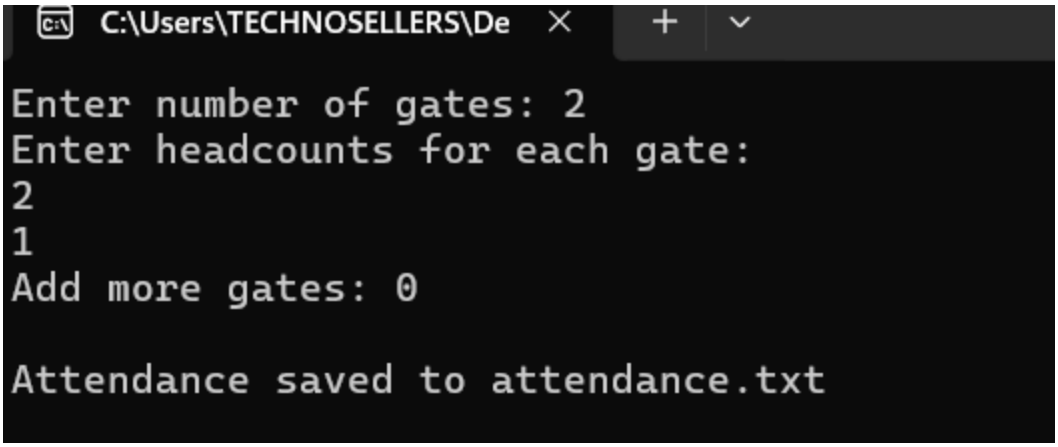
```
    fclose(f);

    printf("\nAttendance saved to attendance.txt\n");

    free(gates);
    return 0;
}

```

Output 6:


C:\Users\TECHNOSELLERS\De
Enter number of gates: 2
Enter headcounts for each gate:
2
1
Add more gates: 0
Attendance saved to attendance.txt

Task 7:

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

typedef struct {
    char medicineName[50];
    int quantityAvailable;
    float unitPrice;
} Medicine;

int main() {
    Medicine *list = NULL;
    int n = 0, choice, i, removeIndex;

    while (1) {
        printf("\n1. Add Medicine\n2. Remove Medicine\n3. Exit & Save\nChoose: ");
        scanf("%d", &choice);

        if (choice == 1) {
            // Add medicine (resize array)
            list = realloc(list, (n + 1) * sizeof(Medicine));

            printf("Medicine Name: ");
            scanf("%s", list[n].medicineName);

            printf("Quantity: ");
            scanf("%d", &list[n].quantityAvailable);

            printf("Unit Price: ");
            scanf("%f", &list[n].unitPrice);

            n++;
```

```
31
32          n++;
33      }
34      else if (choice == 2) {
35          if (n == 0) {
36              printf("No medicines to remove.\n");
37              continue;
38          }
39
40          printf("Enter index to remove (1 to %d): ", n);
41          scanf("%d", &removeIndex);
42          removeIndex--;
43
44          if (removeIndex < 0 || removeIndex >= n) {
45              printf("Invalid index.\n");
46              continue;
47          }
48
49          // Shift elements left
50          for (i = removeIndex; i < n - 1; i++)
51              list[i] = list[i + 1];
52
53          // Shrink array
54          n--;
55          list = realloc(list, n * sizeof(Medicine));
56
57          printf("Medicine removed.\n");
58      }
59      else if (choice == 3) {
60          break;
61      }
62      else {
```

```
        }
        else {
            printf("Invalid choice.\n");
        }
    }

    // Compute totals + low stock
    float totalValue = 0;
    int lowStockCount = 0;

    for (i = 0; i < n; i++) {
        totalValue += list[i].quantityAvailable * list[i].unitPrice;

        if (list[i].quantityAvailable < 5)
            lowStockCount++;
    }

    // Save to file
    FILE *f = fopen("medicine_inventory.txt", "w");

    fprintf(f, "Medicine Inventory Summary\n");
    fprintf(f, "--------------------------\n");

    for (i = 0; i < n; i++) {
        fprintf(f, "%s | Qty: %d | Price: %.2f | Value: %.2f\n",
                list[i].medicineName,
                list[i].quantityAvailable,
                list[i].unitPrice,
                list[i].quantityAvailable * list[i].unitPrice);
    }

    fprintf(f, "\nTotal Inventory Value: %.2f\n", totalValue);
```

```
    FILE *f = fopen("medicine_inventory.txt", "w");

    fprintf(f, "Medicine Inventory Summary\n");
    fprintf(f, "--------------------------\n");

    for (i = 0; i < n; i++) {
        fprintf(f, "%s | Qty: %d | Price: %.2f | Value: %.2f\n",
                list[i].medicineName,
                list[i].quantityAvailable,
                list[i].unitPrice,
                list[i].quantityAvailable * list[i].unitPrice);
    }

    fprintf(f, "\nTotal Inventory Value: %.2f\n", totalValue);
    fprintf(f, "Low-Stock Medicines (<5 qty): %d\n", lowStockCount);

    fclose(f);

    printf("\nData saved to medicine_inventory.txt\n");

    free(list);
    return 0;
}
```

Output 7:

```
1. Add Medicine
2. Remove Medicine
3. Exit & Save
Choose: 1
Medicine Name: panadol
Quantity: 5
Unit Price: 30

1. Add Medicine
2. Remove Medicine
3. Exit & Save
Choose: 2
Enter index to remove (1 to 1): 1
Medicine removed.

1. Add Medicine
2. Remove Medicine
3. Exit & Save
Choose: 3

Data saved to medicine_inventory.txt
```