// Question 1

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

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void display(int arr[], int low, int high)
{
    for (int i = low; i <= high; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int partition(int arr[], int low, int high)
{
    int pivotIndex = low + rand() % (high - low + 1);

    swap(&arr[pivotIndex], &arr[high]);

    int pivot = arr[high];

    printf("\nPivot chosen: %d\n", pivot);
    printf("Before partition: ");
    display(arr, low, high);

    int i = low - 1;

    for (int j = low; j < high; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[high]);

    printf("After partition : ");
    display(arr, low, high);

    return i + 1;
}

void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int p = partition(arr, low, high);
```

```
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}

int main()
{
    int n = 100;
    int marks[n];

    for (int i = 0; i < n; i++)
        marks[i] = rand() % 100;
    srand(time(NULL));

    quickSort(marks, 0, n - 1);

    printf("\nFINAL SORTED MARKS\n");
    for (int i = 0; i < n; i++)
        printf("%d ", marks[i]);

    printf("\n Weak Students (Lowest Marks): %d %d\n",
           marks[0], marks[1]);

    printf("Top Rankers (Highest Marks): %d %d\n",
           marks[n - 2], marks[n - 1]);

    return 0;
}
```

// Question 2

```
#include <stdio.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j < high; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
```

```
}

int quickSelect(int arr[], int low, int high, int k)
{
    if (low <= high)
    {
        int p = partition(arr, low, high);

        if (p == k)
            return arr[p];
        else if (p > k)
            return quickSelect(arr, low, p - 1, k);
        else
            return quickSelect(arr, p + 1, high, k);
    }
    return -1;
}

int main()
{
    int n, k, choice;

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

    int marks[n];

    printf("Enter marks:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &marks[i]);

    printf("Enter value of k: ");
    scanf("%d", &k);

    printf("\n1. k-th Smallest\n2. k-th Largest\nEnter choice: ");
    scanf("%d", &choice);

    if (choice == 1)
    {
        int result = quickSelect(marks, 0, n - 1, k - 1);
        printf("%d-th Smallest Mark = %d\n", k, result);
    }
    else if (choice == 2)
    {
        int result = quickSelect(marks, 0, n - 1, n - k);
        printf("%d-th Largest Mark = %d\n", k, result);
    }
    else
    {
        printf("Invalid choice\n");
    }

    return 0;
}
```

// Question 3

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

struct Patient {
    int id;
    char name[50];
    int age;
    struct Patient *next;
};

struct Patient *head = NULL;

void addPatient() {
    struct Patient *newNode, *temp;
    newNode = (struct Patient *)malloc(sizeof(struct Patient));

    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }

    printf("Enter Patient ID: ");
    scanf("%d", &newNode->id);

    printf("Enter Patient Name: ");
    scanf(" %[^\n]", newNode->name);

    printf("Enter Patient Age: ");
    scanf("%d", &newNode->age);

    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }

    printf("Patient added successfully!\n");
}

void displayPatients() {
    struct Patient *temp = head;

    if (head == NULL) {
        printf("Waiting list is empty.\n");
        return;
    }

    printf("\n--- Waiting Patients ---\n");
    while (temp != NULL) {
        printf("ID: %d | Name: %s | Age: %d\n",
               temp->id, temp->name, temp->age);
```

```
        temp = temp->next;
    }
}

void countPatients() {
    int count = 0;
    struct Patient *temp = head;

    while (temp != NULL) {
        count++;
        temp = temp->next;
    }

    printf("Total patients waiting: %d\n", count);
}

int main() {
    int choice;

    do {
        printf("\n===== Hospital Waiting List Menu =====\n");
        printf("1. Add New Patient\n");
        printf("2. Display Waiting Patients\n");
        printf("3. Count Total Patients\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            addPatient();
            break;
        case 2:
            displayPatients();
            break;
        case 3:
            countPatients();
            break;
        case 4:
            printf("Exiting safely...\n");
            break;
        default:
            printf("Invalid choice! Try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```

// Question 4

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

struct Patient {
    int id;
```

```
    char name[50];
    int age;
    struct Patient *next;
};

struct Patient *head = NULL;

struct Patient* createPatient() {
    struct Patient *newNode =
        (struct Patient*)malloc(sizeof(struct Patient));

    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return NULL;
    }

    printf("Enter Patient ID: ");
    scanf("%d", &newNode->id);

    printf("Enter Patient Name: ");
    scanf(" %[^\n]", newNode->name);

    printf("Enter Patient Age: ");
    scanf("%d", &newNode->age);

    newNode->next = NULL;
    return newNode;
}

void addEmergencyPatient() {
    struct Patient *newNode = createPatient();
    if (newNode == NULL) return;

    newNode->next = head;
    head = newNode;

    printf("Emergency patient added at front!\n");
}

void addRegularPatient() {
    struct Patient *newNode = createPatient();
    if (newNode == NULL) return;

    if (head == NULL) {
        head = newNode;
    } else {
        struct Patient *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }

    printf("Regular patient added at end!\n");
}

void displayPatients() {
    struct Patient *temp = head;
```

```
    if (head == NULL) {
        printf("Waiting list is empty.\n");
        return;
    }

    printf("\n--- Patient Waiting List ---\n");
    while (temp != NULL) {
        printf("ID: %d | Name: %s | Age: %d\n",
               temp->id, temp->name, temp->age);
        temp = temp->next;
    }
}

void removePatient() {
    int id;
    printf("Enter Patient ID to remove: ");
    scanf("%d", &id);

    struct Patient *temp = head, *prev = NULL;

    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    if (head->id == id) {
        head = head->next;
        free(temp);
        printf("Patient removed successfully.\n");
        return;
    }

    while (temp != NULL && temp->id != id) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Patient not found.\n");
        return;
    }

    prev->next = temp->next;
    free(temp);
    printf("Patient removed successfully.\n");
}

void updatePatient() {
    int id;
    printf("Enter Patient ID to update: ");
    scanf("%d", &id);

    struct Patient *temp = head;

    while (temp != NULL && temp->id != id)
        temp = temp->next;

    if (temp == NULL) {
```

```
        printf("Patient not found.\n");
        return;
    }

    printf("Enter new Name: ");
    scanf(" %[^\n]", temp->name);

    printf("Enter new Age: ");
    scanf("%d", &temp->age);

    printf("Patient details updated successfully.\n");
}

void countPatients() {
    int count = 0;
    struct Patient *temp = head;

    while (temp != NULL) {
        count++;
        temp = temp->next;
    }

    printf("Total patients waiting: %d\n", count);
}

int main() {
    int choice;

    do {
        printf("\n===== Hospital Emergency Waiting List =====\n");
        printf("1. Add Emergency Patient (Front)\n");
        printf("2. Add Regular Patient (End)\n");
        printf("3. Display Patients\n");
        printf("4. Remove Patient\n");
        printf("5. Update Patient Details\n");
        printf("6. Count Patients\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            addEmergencyPatient();
            break;
        case 2:
            addRegularPatient();
            break;
        case 3:
            displayPatients();
            break;
        case 4:
            removePatient();
            break;
        case 5:
            updatePatient();
            break;
        case 6:
            countPatients();
```

```
            break;
        case 7:
            printf("Exiting safely...\n");
            break;
        default:
            printf("Invalid choice! Try again.\n");
        }
    } while (choice != 7);

    return 0;
}
```

// Question 5

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

struct Node {
    int data;
    struct Node *next;
};

struct Node* createNode(int data) {
    struct Node *n = (struct Node*)malloc(sizeof(struct Node));
    n->data = data;
    n->next = NULL;
    return n;
}

struct Node* destroyUnstable(struct Node *head, int K) {
    if (K <= 1 || head == NULL) return head;

    struct Node dummy;
    dummy.next = head;

    struct Node *prev = &dummy;
    struct Node *start = head;

    while (start != NULL) {
        struct Node *temp = start;
        int count = 0;

        int maxVal = temp->data, minVal = temp->data;
        int maxPos = 0, minPos = 0, idx = 0;

        while (temp != NULL && count < K) {
            if (temp->data > maxVal) {
                maxVal = temp->data;
                maxPos = idx;
            }
            if (temp->data < minVal) {
                minVal = temp->data;
                minPos = idx;
            }
            temp = temp->next;
            idx++;
```

```
            count++;
        }

        if (count < K) break;

        if (maxPos < minPos) {
            temp = start;
            for (int i = 0; i < K; i++) {
                struct Node *del = temp;
                temp = temp->next;
                free(del);
            }

            prev->next = temp;
            start = temp;
        } else {
            prev = start;
            start = start->next;
        }
    }

    return dummy.next;
}

void printList(struct Node *head) {
    if (head == NULL) {
        printf("-1\n");
        return;
    }
    printf("%d\n", head->data);
}

int main() {
    int n, k, x;
    struct Node *head = NULL, *tail = NULL;

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

    for (int i = 0; i < n; i++) {
        scanf("%d", &x);
        struct Node *node = createNode(x);
        if (head == NULL)
            head = tail = node;
        else {
            tail->next = node;
            tail = node;
        }
    }

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

    head = destroyUnstable(head, k);
    printList(head);

    return 0;
}
```