Assignment - 4

Q1

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

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

void display partition (int arr[], int low, int high) {

    int pivotIndex = low + rand()% (high - low + 1);

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

    int pivot = arr[high];

printf ("\n Pivot 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 display (int arr[], int low, int high) {

    for (int i= low; i<= high; i++) {

        printf("%d", arr[i]);
    }
    printf("\n");
}

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("\n Final 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;
}
```

Q2

```
#include <stdio.h>

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

int partision (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 = partision (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("\n 1. k-th smallest \n 2. k-th largest \n
        Enter 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 ("Invald choice \n");
    }

return 0;

}
```

Q3

```
#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);

newNode -> next = NULL;
```

```
Void displayPatients() {
    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.\n");
    } while (choice != 4);

    return 0;
}
```

Q4

```
#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 (size of (struct Patient));

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

    }

Printf ( " Enter data : ");
scanf ( " %d ", & newNode -> id);
scanf ( " %[^\n]", newNode -> name );
scanf ( " %d", & newNode -> age);

    newNode -> next = NULL;
    return newNode;
}

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;

    }

}
```

```
void displayPatients() {
    struct Patient * temp = head;
    if (head == NULL) {
        printf("Waiting list empty.\n");
        return;
    }
}

void removePatients() {
    int id;
    scanf("%d", &id);
    struct Patient * temp = head;
    while (temp != NULL && temp->id != id)
        printf("Patient not found.\n");
        return;
}

Void countPatients() {
    int count = 0;
    struct Patient * temp = head;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
}
```

```
int main() {
    int choice;

do {    scanf("%d", &choice);

    switch (choice) {
        case 1:
            addEmergencyPatient();
            break;
        case 2:
            addregularpatient();
            break;
        case 3:
            displaypatient();
            break;
        case 4:
            removepatient();
            break;
        case 5:
            UpdatePatient();
            break;
        case 6:
            CountPatient();
        case 7:
            printf("Exiting...");
            break;
        default:
            Printf("Invalid...");
    } while (choice != 7);

    return 0;
}
```

Q5

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

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

struct Node * create(int data) {
    struct Node * n = (struct Node*) malloc
    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;
```

```
Void Print list ( 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 no. of Nodes : ");
    Scanf("%d", &n);

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

    Scanf("%d", &x) ;

    Struct Node * node = Create Node (x);
    if (head = = NULL)
            head = tail = node;
    else
            tail -> next = node;
            tail = node;

}
```

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

head = destroy_unstable(head, k);
printlist(head);

return 0;

}
```