Assignment-2

Q. Implement radix sort using counting sort

```
 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <limits.h>

int findMax(int array[], int h){

   int max=array[0];
   for(int j=1; j<h; j++){
      if(max<array[j]){
         max=array[j];
      }
      else{
         continue;
      }
   }
   return max;
}

int findDigits(int max) {
   int digit = 0;
   while (max != 0) {
      max = max / 10;
      digit++;
```

```
    }
    return digit;
}

int main(){

    int n=6;
    int radix[6];
    int radix2[n];
    printf("Specify %d elements\n", n);
    for(int i=0; i<n; i++){
        scanf("%d", &radix[i]);
    }

    int max= findMax(radix, n);

    int digit = findDigits(max);
    int it=1;
    int exp=1;
    int countarray[10];
    while(digit>0){
        for(int i=0; i<10; i++){
            countarray[i]=0;
        }
        for(int i=0; i<n; i++){
            countarray[(radix[i]/exp)%10]++;
        }
        for(int i=1; i<=10; i++){
            countarray[i] = countarray[i] + countarray[i-1];
        }
        for (int i = n - 1; i >= 0; i--) {
```

```
        int currentDigit = (radix[i] / exp) % 10;
        radix2[countarray[currentDigit] - 1] = radix[i];
        countarray[currentDigit]--;
    }
        for(int i=0; i<n; i++){
            radix[i]=radix2[i];
        }
        digit--;
        exp=exp*10;

        printf("\nIteration: %d\n", it);
        for(int j=0; j<n; j++){
            printf("%d\t", radix[j]);
        }
        it++;
    }
    printf("\nSorted array:\n");
    for(int j=0; j<n; j++){
        printf("%d\t", radix[j]);
    }
    return 0;
}
```

```
Specify 6 elements
84
521
487631
1234
1
874

Iteration: 1
521     487631  1       84      1234    874
Iteration: 2
1       521     487631  1234    874     84
Iteration: 3
1       84      1234    521     487631  874
Iteration: 4
1       84      521     874     1234    487631
Iteration: 5
1       84      521     874     1234    487631
Iteration: 6
1       84      521     874     1234    487631
Sorted array:
1       84      521     874     1234    487631
Process returned 0 (0x0)   execution time : 15.483 s
Press any key to continue.
```

Q. Implement radix sort using

```
#include <stdio.h>

#include <stdlib.h>


int checkDigit(int number, int max) {

   int count = 0;


   do {

      count++;

      number /= 10;

   } while (number != 0);


   if(max < count) {
```

```
        max = count;
    }
    return max;
}

struct radix {
    int ele;
    struct radix *next;
};

struct radixhead {
    int number;
    struct radix *list;
    struct radixhead *next;
};

void appendToBucket(struct radixhead **bucket, int digit, int number) {
    struct radixhead *travel = *bucket;
    struct radixhead *prev = NULL;

    while(travel != NULL && travel->number < digit) {
        prev = travel;
        travel = travel->next;
    }

    if(travel == NULL || travel->number != digit) {
        struct radixhead *newBucket = (struct radixhead*)malloc(sizeof(struct radixhead));
        newBucket->number = digit;
        newBucket->list = NULL;
        newBucket->next = travel;
```

```
        if(prev == NULL) {
            *bucket = newBucket;
        } else {
            prev->next = newBucket;
        }
        travel = newBucket;
    }

    struct radix *newElement = (struct radix*)malloc(sizeof(struct radix));
    newElement->ele = number;
    newElement->next = NULL;

    if(travel->list == NULL) {
        travel->list = newElement;
    } else {
        struct radix *listTravel = travel->list;
        while(listTravel->next != NULL) {
            listTravel = listTravel->next;
        }
        listTravel->next = newElement;
    }
}

void freeBuckets(struct radixhead *bucket) {
    struct radixhead *tempBucket;
    while(bucket != NULL) {
        struct radix *tempList;
        while(bucket->list != NULL) {
            tempList = bucket->list;
            bucket->list = bucket->list->next;
            free(tempList);
```

```
        }
        tempBucket = bucket;
        bucket = bucket->next;
        free(tempBucket);
    }
}

struct radix* collectAndSort(struct radixhead *bucket, int *isEmpty) {
    struct radix *head = NULL;
    struct radix *tail = NULL;

    while(bucket != NULL) {
        struct radix *listTravel = bucket->list;
        while(listTravel != NULL) {
            struct radix *newElement = (struct radix*)malloc(sizeof(struct radix));
            newElement->ele = listTravel->ele;
            newElement->next = NULL;

            if(head == NULL) {
                head = newElement;
                tail = newElement;
            } else {
                tail->next = newElement;
                tail = newElement;
            }

            listTravel = listTravel->next;
        }
        bucket = bucket->next;
    }
```

```
    *isEmpty = (head == NULL);
    return head;
}

void radixSort(struct radix **head, int maxdigit) {
    int exp = 1;
    for(int i = 0; i < maxdigit; i++) {
        struct radixhead *bucket = NULL;
        struct radix *sort = *head;
        while(sort != NULL) {
            int digit = (sort->ele / exp) % 10;
            appendToBucket(&bucket, digit, sort->ele);
            sort = sort->next;
        }

        int isEmpty = 0;
        *head = collectAndSort(bucket, &isEmpty);
        freeBuckets(bucket);

        exp *= 10;
        if(isEmpty) break;
    }
}

void printList(struct radix *head) {
    struct radix *temp = head;
    while(temp != NULL) {
        printf("%d ", temp->ele);
        temp = temp->next;
    }
    printf("\n");
```

```
}

int main() {
    int maxdigit = 0;
    struct radix *head = NULL;
    struct radix *temp = NULL;

    printf("Enter elements to sort\n-1 to end\n");
    int c = 0;
    while(c >= 0) {
        scanf("%d", &c);
        if(c < 0) break;

        struct radix *block = (struct radix*)malloc(sizeof(struct radix));
        block->ele = c;
        block->next = NULL;

        if(head == NULL) {
            head = block;
            temp = head;
        } else {
            temp->next = block;
            temp = block;
        }

        maxdigit = checkDigit(c, maxdigit);
    }

    printf("Before sorting: ");
    printList(head);
```

```
    radixSort(&head, maxdigit);

    printf("After sorting: ");
    printList(head);

    return 0;
}
```

```
C:\Users\Tarun\ENGG\SEASOI

Enter elements to sort
-1 to end
87
654
412
782
425
487
321
-1
Before sorting: 87 654 412 782 425 487 321
After sorting: 87 321 412 425 487 654 782

Process returned 0 (0x0)   execution time : 18.670 s
Press any key to continue.
```