**V.Venkateswarlu Gupta**

**1602-19-733-119**

**CSE-B**

**Data Structures**

*Assignment – 1*

**1. Given an array of integers, Develop a C program to replace every element with the next greatest element (greatest element on the right side) in the array. Since there is no element next to the last element, replace it with -1. For example, if the array is {16, 17, 4, 3, 5, 2}, then it should be modified to {17, 5, 5, 5, 2, -1}**

**Input: 16, 17, 4, 3, 5, 2**

**Output:17, 5, 5, 5, 2, -1**

**A.**

#include<stdio.h>

*void* main()

{

*int* a[50],i,j,n;

    printf("Enter no of terms : ");

    scanf("%d",&n);

    printf("Enter elements\n");

    for(i=0;i<n;i++)

        scanf("%d",&a[i]);

    for(i=0;i<n-1;i++)

    {

*int* m=0;

        for(j=i+1;j<n;j++)

        {

            if(a[j]>m)

                m=a[j];

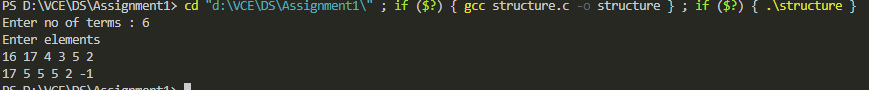
        }

        printf("%d ",m);

    }

    printf("-1");

}



**2. Given employee details employee id, name, department and basic salary. Develop a C program to display employee details department wise in the alphabetical order. Also design the function to compute the total number of employees in every department and determine the time complexity for the designed function. Use linked list to implement the solution.**

**A.**

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

*char* depts[6][6]={"civil","cse","ece","eee","it","mech"};

*int* deptcnt[6]={0};

typedef *struct* E

{

*int* id;

*char* name[32];

*char* dept[8];

*float* salary;

*struct* E \*link;

}E;

*void* addElements(E \**h*)

{

*int* n;

    printf("No of elements : ");

    scanf("%d",&n);

    E \*q=NULL;

    q=*h*;

    if(q->id==-1)

    {

        for(*int* i=0;i<n-1;i++)

        {

            E \*t=(E \*)calloc(1,sizeof(E));

            scanf("%d%s%s%f",&q->id,q->name,q->dept,&q->salary);

            q->link=t;

            q=t;

        }

        scanf("%d%s%s%f",&q->id,q->name,q->dept,&q->salary);

        q->link=NULL;

    }

    else

    {

        while(q->link!=NULL)

        {

            q=q->link;

        }

        for(*int* i=0;i<n;i++)

        {

            E \*t=(E \*)calloc(1,sizeof(E));

            q->link=t;

            q=t;

            scanf("%d%s%s%f",&q->id,q->name,q->dept,&q->salary);

        }

        q->link=NULL;

    }

}

*void* bubbleStrings(E \**h*)

{

    E \*i,\*j;

    i=*h*;

    while(i->link!=NULL)

    {

        j=i->link;

        while(j->link!=NULL)

        {

            if(strcmp(i->name,j->name)>0)

            {

*int* tempid;

*float* tempsalary;

*char* tempname[32],tempdept[6];

                strcpy(tempname,i->name);

                strcpy(i->name,j->name);

                strcpy(j->name,tempname);

                strcpy(tempdept,i->dept);

                strcpy(i->dept,j->dept);

                strcpy(j->dept,tempdept);

                tempid=i->id;

                i->id=j->id;

                j->id=tempid;

                tempsalary=i->salary;

                i->salary=j->salary;

                j->salary=tempsalary;

            }

            j=j->link;

        }

        if(strcmp(i->name,j->name)>0)

        {

*int* tempid;

*float* tempsalary;

*char* tempname[32],tempdept[6];

                strcpy(tempname,i->name);

                strcpy(i->name,j->name);

                strcpy(j->name,tempname);

                strcpy(tempdept,i->dept);

                strcpy(i->dept,j->dept);

                strcpy(j->dept,tempdept);

                tempid=i->id;

                i->id=j->id;

                j->id=tempid;

                tempsalary=i->salary;

                i->salary=j->salary;

                j->salary=tempsalary;

        }

        i=i->link;

    }

}

*void* displaydept(E \**h*)

{

    E \*q=NULL;

*int* i=0;

    while(i<6)

    {

        q=*h*;

        while(q->link!=NULL)

        {

            if(!strcmp(depts[i],q->dept))

            {

                printf("%3d   %7s   %4s        %4.2f\n",q->id,q->name,q->dept,q->salary);

                deptcnt[i]++;

            }

            q=q->link;

        }

        if(!strcmp(depts[i],q->dept))

        {

            printf("%3d   %7s   %4s        %4.2f\n",q->id,q->name,q->dept,q->salary);

            deptcnt[i]++;

        }

        i++;

    }

}

*void* main()

{

    E \*h=(E \*)calloc(1,sizeof(E));

    h->id=-1;

    addElements(h);

    bubbleStrings(h);

    printf("\n\n Displaying the details of emploees \n");

    printf("Id      Name    Department  Salary\n");

    displaydept(h);

    printf("No of Employees in each department\n");

    for(*int* i=0;i<6;i++)

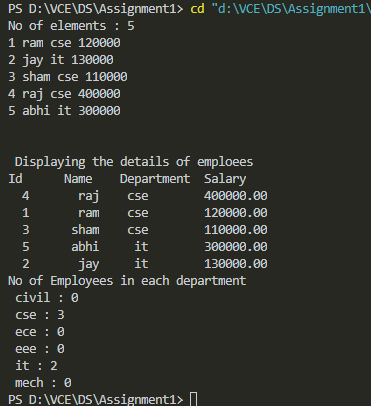
    {

        // printf("%S : ",depts[i]);

        printf(" %s : %d \n",depts[i],deptcnt[i]);

    }

}



**3. Write a C program to include append( ) function that takes two linked lists, a and b, appends b to the end of a, and returns the head of the new list.**

**Input: a->b->a->c**

**d->a->m**

**Output: a->b->a->c->d->a->m**

**Input: Null**

**d->n**

**Output: d->n**

**A.**

#include<stdio.h>

#include<stdlib.h>

typedef *struct* node{

*char* c;

*struct* node \*link;

}node;

*void* create(node \**h*){

*int* i,n;

    node \*q;

    q=*h*;

    printf("No of elements : ");

    scanf("%d",&n);

    if(n==0) return;

    for(i=0;i<n-1;i++){

        if(q->c=='#'){

            scanf(" %c",&q->c);

        }

        node \*t=(node \*)malloc(sizeof(node));

        scanf("%c",&t->c);

        t->link=NULL;

        q->link=t;

        q=t;

    }

}

node \*append(node \**h1*,node \**h2*){

    if(*h1*->c=='#'&&*h2*->c=='#')

        return *h1*;

    else if(*h1*->c=='#'){

        return *h2*;

    }

    else if(*h2*->c=='#'){

        return *h1*;

    }

    else{

        node \*q;

        q=*h1*;

        while(q->link!=NULL){

            q=q->link;

        }

        q->link=*h2*;

        return *h1*;

    }

}

*void* display(node \**h*){

    node \*q;

    q=*h*;

    if(q->c=='#') return;

    while(q->link!=NULL){

        printf("%c->",q->c);

        q=q->link;

    }

    printf("%c\n",q->c);

}

*void* main(){

    node \*h1,\*h2,\*h;

    h1=(node \*)malloc(sizeof(node));

    h1->c='#';

    h1->link=NULL;

    create(h1);

    h2=(node \*)malloc(sizeof(node));

    h2->c='#';

    h2->link=NULL;

    create(h2);

    h=append(h1,h2);

    display(h);

}

