**V Venkateswarlu Gupta**

**1602-19-733-119**

**CSE-B**

**Data Structures**

*Assignment – 2*

1. **Write a program to implement stack using queue.**

#include<stdio.h>

#include<stdlib.h>

*int* front=-1;

*int* rear=-1;

#define n 50

//performing rotation after every enQ(push) operation into the Q

*int* a[n],cnt=0;

*void* rotate(){

*int* t=a[cnt-1];

    for(*int* i=cnt-1;i>0;i--){

        a[i]=a[i-1];

    }

    a[0]=t;

}

*void* enqueue(){

*int* x;

    printf("Enter element : ");

    scanf("%d",&x);

    if(rear==n-1){

        printf("Queue is full\n");

        return;

    }else if(front==-1 && rear==-1){

        front++;

    }

    ++cnt;

    a[++rear]=x;

    rotate();

}

*int* dequeue(){

*int* x;

    if(front==-1 && rear==-1){

        printf("No elements to delete\n");

    }else if(front==rear){

        front=-1;

        rear=-1;

    }else{

        x=a[front];

        front++;

        cnt--;

        return x;

    }

}

*void* display(){

    printf("Elements in the Stack\n");

    if(front==-1){

        printf("Queue is empty\n");

    }else{

        for(*int* i=front;i<rear;i++){

        printf("%d->",a[i]);

        }

    }

    printf("%d\n",a[rear]);

}

*void* main()

{

*int* x;

*short* repeat;

    do

    {

*short* choice;

        printf(" 1 push\t 2 pop\t 3 display\t # exit : ");

        scanf("%hi",&choice);

        switch(choice)

        {

            case 1:enqueue();

                   break;

            case 2:x=dequeue();

                   printf("%d is poped\n",x);

                   break;

            case 3:display();

                   break;

            default:exit(1);

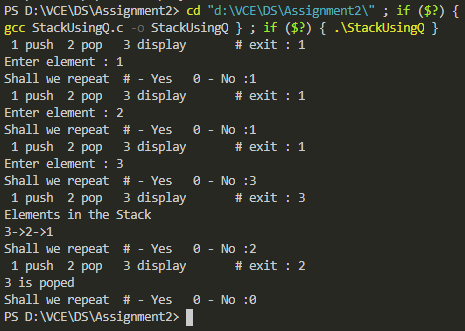
        }

        printf("Shall we repeat\t # - Yes   0 - No :");

        scanf("%hi",&repeat);

    }while(repeat);

}



**2. Given a circular linked list, we need to arrange the consonants and vowel nodes of it in such a way that all the vowels nodes come before the consonants while maintaining the order of their arrival.**

**Examples:**

**Input :**

**a -> b -> c -> e -> d -> o -> x -> i**

**Output :**

**a -> e -> o -> i -> b -> c -> d -> x**

#include<stdlib.h>

#include<stdio.h>

typedef *struct* node{

*char* ch;

*struct* node \*link;

}node;

node \*h,\*h1;

*void* create(){

*int* n;

    node \*q,\*q1,\*j;

    q=h;

    q1=h1;

    j=h1;

    printf("Enter no of letters : ");

    scanf("%d",&n);

    for(*int* z=0;z<n;z++){

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

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

        scanf(" %c",&t->ch);

        t->link=NULL;

        tt->link=NULL;

        if(h==NULL){

            h=t;

            q=t;

            h1=tt;

            q1=tt;

        }else{

            q->link=t;

            q=q->link;

            q1->link=tt;

            q1=q1->link;

        }

    }

    q->link=h;

    q1->link=h1;

}

*void* arrange(){

    node \*q,\*q1;

    q=h;

    q1=h1;

    while(q->link!=h){

        if(q->ch=='a'||q->ch=='e'||q->ch=='i'||q->ch=='o'||q->ch=='u'){

            q1->ch=q->ch;

            q1=q1->link;

        }

        q=q->link;

    }

    if(q->ch=='a'||q->ch=='e'||q->ch=='i'||q->ch=='o'||q->ch=='u'){

        q1->ch=q->ch;

        q1=q1->link;

    }

    q=h;

    while(q->link!=h){

        if(q->ch!='a'&&q->ch!='e'&&q->ch!='i'&&q->ch!='o'&&q->ch!='u'){

            q1->ch=q->ch;

            q1=q1->link;

        }

        q=q->link;

    }

    if(q->ch!='a'&&q->ch!='e'&&q->ch!='i'&&q->ch!='o'&&q->ch!='u'){

        q1->ch=q->ch;

        q1=q1->link;

    }

}

*void* display(node \**head*){

    node \*q;

    q=*head*;

*int* i=0;

    //while(q->link!=h)

    while(i<14)

    {

        printf("%c->",q->ch);

        q=q->link;

        i++;

    }

    printf("%c\n",q->ch);

}

*void* main(){

    h=NULL;

    h1=NULL;

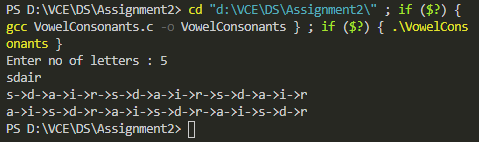
    create();

    arrange();

    display(h);

    display(h1);

}



**To ensure its circular list, I have printed more than once**

**3. Write a program to find the transpose of a sparse matrix using linked list.**

#include<stdio.h>

#include<stdlib.h>

typedef *struct* node{

*int* val;

*int* row;

*int* col;

*struct* node \*link;

}Sparse;

Sparse \*h;

*int* cnt=0,m,n;

*void* storeFromFile(){

    FILE \*fp;

    Sparse \*q;

    q=h;

    fp=fopen("input.txt","r");

*int* i,j,v;

    fscanf(fp,"%d %d",&m,&n);

    for(i=0;i<m;i++){

        for(j=0;j<n;j++){

            fscanf(fp,"%d",&v);

            if(v!=0){

                cnt++;

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

                t->val=v;

                t->row=i;

                t->col=j;

                t->link=NULL;

                if(q==NULL){

                    h=t;

                    q=t;

                }else{

                    q->link=t;

                    q=t;

                }

            }

        }

    }

}

*void* display(){

    Sparse \*q;

    q=h;

*int* i=0;

    if(q==NULL){

        printf("All elements are ZERO's\n");

    }else if(q->link==NULL){

        printf("ROW \t COLUMN \t VALUE\n");

        printf(" %d  \t    %d  \t  \t %d\n",q->row,q->col,q->val);

    }else{

        printf("ROW \t COLUMN \t VALUE\n");

        while(q!=NULL){

            printf(" %d  \t    %d  \t  \t %d\n",q->row,q->col,q->val);

            q=q->link;

            i++;

        }

    }

}

Sparse \*copy(*int* *rowcol*){//if rowcol is one, then rows and columns will be copied oppositely

    Sparse \*t,\*q,\*qc,\*hc;

    q=h;

    hc=NULL;

    qc=hc;

    while(q!=NULL){

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

        t->link=q->link;

        if(*rowcol*==0){

            t->row=q->row;

            t->col=q->col;

        }else if(*rowcol*==1){

            t->row=q->col;

            t->col=q->row;

        }

        t->val=q->val;

        if(hc==NULL){

            hc=t;

            qc=t;

        }else{

            qc->link=t;

            qc=t;

        }

        q=q->link;

    }

    return hc;

}

*void* bubbleSort(Sparse \**h*){

    Sparse \*i,\*j;

    i=*h*;

    while(i!=NULL)

    {

        j=i->link;

        while(j!=NULL)

        {

            if(i->row>j->row)

            {

                i->row=i->row+j->row;

                j->row=i->row-j->row;

                i->row=i->row-j->row;

                i->col=i->col+j->col;

                j->col=i->col-j->col;

                i->col=i->col-j->col;

                i->val=i->val+j->val;

                j->val=i->val-j->val;

                i->val=i->val-j->val;

            }

            j=j->link;

        }

        i=i->link;

    }

}

*void* storeToFile(Sparse \**r*,*int* *m0*,*int* *n0*,*char* \**filename*){

    FILE \*fp;

    Sparse \*q;

    q=*r*;

    fp=fopen(*filename*,"w");

    fprintf(fp,"%d %d\n",*m0*,*n0*);

    for(*int* j,i=0;i<*m0*;i++){

        for(j=0;j<*n0*;j++){

            if(q->row==i&&q->col==j){

                fprintf(fp,"%d ",q->val);

                q=q->link;

            }else{

                fprintf(fp,"0 ");

            }

        }

        fprintf(fp,"\n");

    }

}

Sparse \*transpose(){

    Sparse \*q,\*ht=copy(1);

    q=ht;

    printf("Transpose of the matrix\n");

    bubbleSort(ht);

    display(ht);

    storeToFile(ht,n,m,"transpose.txt");

}

*void* main(){

    h=NULL;

    storeFromFile();

    printf("First Sparse Matrix\n");

    display(h);

    Sparse \*ht=transpose(h);

}

