**V.Venkateswarlu Gupta**

**1602-19-733-119**

**CSE-B**

**Data Structures Lab**

Week – 4

***Prelab Questions***

**1. What are the advantages of linked list over arrays?**

A. Linked List, Arrays are used for storing elements, both use different techniques. In an array, elements are one after the another (successive memory allocation). But in linked list, memory is not contiguous.

**Advantages** :

* Size is not an issue as compared to arrays.
* Addition/Deletion of an element from the list at any index which is an O(1) operation in Lists as compared to Arrays.
* It can grow , shrink it means it has variable size, while size of array is fixed.

**2. What is the need for introducing linked lists?**

A. Linked lists reduces the access time, they are dynamic data structure, resizable at runtime, store elements at different memory locations if the memory is not contiguous, deletion-insertion operations are efficient.

**3. Represent a single linked list with the help of an example.**

A. A linked list has two parts :- Data and next, data can be primitive or non primitive and next consists of address of next node.Head always points to the first nodes address.

HEAD

C NULL

A 100

B 200

Data

Next 100 200

**4.Pseudocode to delete the last element from a linked list.**

*void* delete\_AtTheEnd(node \**h*)

{

    node \*q=NULL;

    q=*h*;

    while(q->link->link!=NULL)

    {

        q=q->link;

    }

    node \*t=q->link;

    q->link=NULL;

    free(t);

}

**5.Write 3 applications of stack.**

A.1.Recursion

2.OS sequence of instructions.

3.pile of plates.

**6. What are the disadvantages of linked lists.**

* More memory is required to store elements in linked list as compared to array. Because in linked list each node contains a pointer and it requires extra memory for itself.
* Elements or nodes traversal is difficult in linked list. We can not randomly access any element as we do in array by index. For example, if we want to access a node at position n then we have to traverse all the nodes before it. So, time required to access a node is large.
* In linked list reverse traversing is difficult.

**7.What condition should be checked when you insert elements onto a stack using arrays.**

A. OVERFLOW condition has to be checked before inserting elements onto a stack. Stack overflow is an undesirable condition in which a program tries to use more memory space than the stack has available.

**8.Write abstract datatypes in stack.**

A.

* Create a ***stack***
* Insert an element on the stack
* Delete an element in a stack
* To check whether stack is empty
* To check which element is on the top of the stack.

**9.Write the procedure to find the mid element of a linked list in a single traversal.**

A. Using two pointers, one moves one node at a time, another moves two nodes at a time. For even no of nodes, using a counter variable we can get mid elements

***Prelab programs***

1. **Operations on linked lists.**

#include<stdio.h>

#include<stdlib.h>

typedef *struct* node

{

*int* data;

*struct* node \*link;

}node;

*void* addElements(node \**h*)

{

*int* n;

    printf("No of elements : ");

    scanf("%d",&n);

    node \*q=NULL;

    q=*h*;

    if(q->data==-37678)

    {

        for(*int* i=0;i<n-1;i++)

        {

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

            scanf("%d",&q->data);

            q->link=t;

            q=t;

        }

        scanf("%d",&q->data);

        q->link=NULL;

    }

    else

    {

        while(q->link!=NULL)

        {

            q=q->link;

        }

        for(*int* i=0;i<n;i++)

        {

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

            q->link=t;

            q=t;

            scanf("%d",&q->data);

        }

        q->link=NULL;

    }

}

node \* insert\_Atbegin(node \**h*)

{

    printf("Enter element to be inserted at the beginning : ");

    if(*h*->data==-37678)

    {

        scanf("%d",&*h*->data);

*h*->link=NULL;

    }

    else

    {

        node \*t;

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

        scanf("%d",&t->data);

        t->link=*h*;

*h*=t;

    }

    return *h*;

}

*void* insert\_InBetween(node \**h*)

{

*int* pos,i=-1;

    node \*q=NULL,\*t=(node \*)malloc(sizeof(node));

    q=*h*;

    printf("Enter position (aray indexing) : ");

    scanf("%d",&pos);

    printf("Enter element to be inserted : ");

    scanf("%d",&t->data);

    while(i<pos-2)

    {

        q=q->link;

        i++;

    }

    t->link=q->link;

    q->link=t;

}

*void* insert\_AtEnd(node \**h*)

{

    printf("Enter element to be inserted at the end : ");

    if(*h*->data=-37678)

    {

        scanf("%d",&*h*->data);

*h*->link=NULL;

    }

    else

    {

        node \*q=NULL,\*t=(node \*)malloc(sizeof(node));

        q=*h*;

        while(q->link!=NULL)

        {

            q=q->link;

        }

        scanf("%d",&t->data);

        q->link=t;

        t->link=NULL;

    }

}

*void* display(node \**h*)

{

    node \*q=NULL;

    q=*h*;

    while(q->link!=NULL)

    {

        printf("%d->",q->data);

        q=q->link;

    }

    printf("%d\n",q->data);

}

*void* count(node \**h*)

{

*int* i=0;

    node \*q=NULL;

    q=*h*;

    while(q->link!=NULL)

    {

        q=q->link;

        i++;

    }

    printf("No of elements : %d\n",++i);

}

node \*delete\_AtBegin(node \**h*)

{

    if(*h*->data==-37678)

        printf("No elements to delete");

    else

    {

        node \*t;

        t=*h*->link;

        free(*h*);

*h*=t;

    }

    return *h*;

}

*void* delete\_InBetween(node \**h*)

{

*int* pos,i=0;

    node \*q=NULL;

    printf("Enter position to delete (array indexing) : ");

    scanf("%d",&pos);

    q=*h*;

    while(i<pos-1)

    {

        q=q->link;

        i++;

    }

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

    t=q->link;

    q->link=q->link->link;

    free(t);

}

*void* delete\_AtTheEnd(node \**h*)

{

    node \*q=NULL;

    q=*h*;

    while(q->link->link!=NULL)

    {

        q=q->link;

    }

    node \*t=q->link;

    q->link=NULL;

    free(t);

}

*void* main()

{

*int* i;

*short* *int* choice,repeat;

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

    h->link=NULL;

    h->data=-37678;

    printf("sInGlLy lInKeD lIsT oPeRaTiOnS\n");

    printf(" 1 Create Singlly Linked List\n");

    printf(" 2 Insertion At the Beginning\n");

    printf(" 3 Insertion In Between\n");

    printf(" 4 Insertion At the End\n");

    printf(" 5 Deletion At the Beginning\n");

    printf(" 6 Deletion In Between\n");

    printf(" 7 Deletion At the End\n");

    printf(" # Exit\n");

    do

    {

        printf("Enter choice : ");

        scanf("%hi",&choice);

        switch(choice)

        {

            case 1:addElements(h);

                   display(h);

                   count(h);

                   break;

            case 2:h=insert\_Atbegin(h);

                   display(h);

                   count(h);

                   break;

            case 3:insert\_InBetween(h);

                   display(h);

                   count(h);

                   break;

            case 4:insert\_AtEnd(h);

                   display(h);

                   count(h);

                   break;

            case 5:h=delete\_AtBegin(h);

                   display(h);

                   count(h);

                   break;

            case 6:delete\_InBetween(h);

                   display(h);

                   count(h);

                   break;

            case 7:delete\_AtTheEnd(h);

                   display(h);

                   count(h);

                   break;

            default:exit(1);

        }

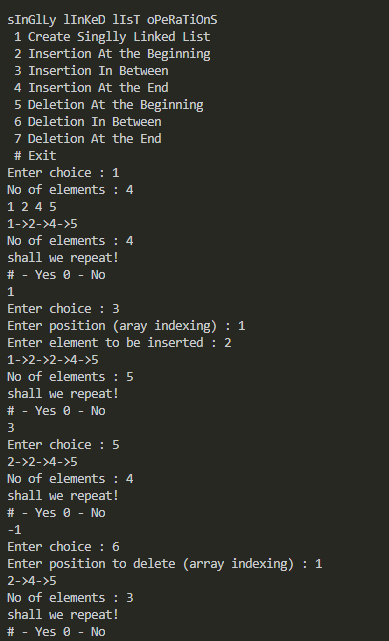
        printf("shall we repeat!\n");

        printf("# - Yes\t0 - No\n");

        scanf("%hi",&repeat);

    }while(repeat);

}



**2.Stacks using arrays.**

#include<stdio.h>

#include<stdlib.h>

*int* top=-1;

*void* push(*int* *size*,*int* \**stack*)

{

    if(top==*size*-1)

        printf("Stack is full, can't push\n");

    else

    {

*int* a;

        ++top;

        printf("Enter element : ");

        scanf("%d",&a);

*stack*[top]=a;

    }

}

*void* pop(*int* *size*,*int* \**stack*)

{

    if(top==-1)

        printf("Stack is empty, no element to delete\n");

    else

        --top;

}

*void* display(*int* *size*,*int* \**stack*)

{

    if(top==-1)

        printf("Stack is empty\n");

    else

    {

        printf("Elements in stack\n");

        for(*int* i=top;i>-1;i--)

            printf("%d ",*stack*[i]);

        printf("\n");

    }

}

*void* main()

{

*int* \*stack,n,repeat;

    printf("Size of stack : ");

    scanf("%d",&n);

    stack=(*int* \*)calloc(n,sizeof(*int*));

    do

    {

        printf(" 1 push\t 2 pop  # exit\n");

*int* choice;

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:push(n,stack);

                   display(n,stack);

                   break;

            case 2:pop(n,stack);

                   display(n,stack);

                   break;

            default:exit(1);

        }

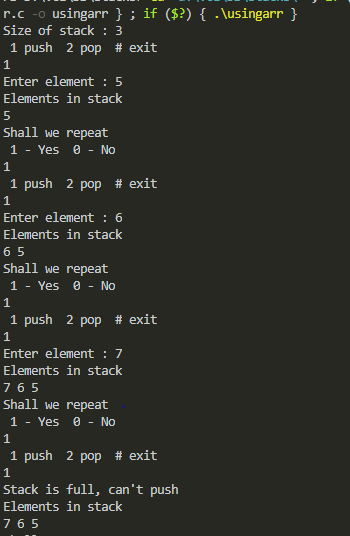
        printf("Shall we repeat\n");

        printf(" 1 - Yes  0 - No\n");

        scanf("%d",&repeat);

    } while(repeat);

}



**Lab Program**

**Stacks using linked lists.**

#include<stdio.h>

#include<stdlib.h>

typedef *struct* node

{

*int* data;

*struct* node \*link;

}node;

node \*h;

*void* push()

{

    node \*t;

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

    printf("Enter element : ");

    scanf("%d",&t->data);

    t->link=NULL;

    if(h==NULL)

        h=t;

    else

    {

        t->link=h;

        h=t;

    }

}

*void* pop()

{

    node \*t;

    if(h==NULL)

        printf("No elements to pop!\n");

    else if(h->link==NULL)

    {

        printf("poping %d\n",h->data);

        t=h;

        h=NULL;

    }

    else

    {

        printf("poping %d\n",h->data);

        t=h;

        h=h->link;

    }

    free(t);

}

*void* display()

{

    node \*t;

    t=h;

    if(h==NULL)

        printf("Stack is empty\n");

    while(t->link!=NULL)

    {

        printf("%d->",t->data);

        t=t->link;

    }

    printf("%d\n",t->data);

}

*void* main()

{

*int* choice,repeat;

    do

    {

        printf(" 1 push\t 2 pop \t3 display   # exit :: ");

*int* choice;

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:push();

                   break;

            case 2:pop();

                   break;

            case 3:display();

                   break;

            default:exit(1);

        }

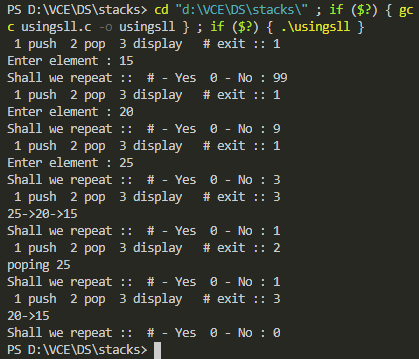
        printf("Shall we repeat :: ");

        printf(" # - Yes  0 - No : ");

        scanf("%d",&repeat);

    }while(repeat);

}



**Q. Bank create, transactions, removal**

#include<stdio.h>

#include<stdlib.h>

typedef *struct* bank{

*short* no;

*char* name[32];

*char* bnm[8];

*int* bal;

*struct* bank \*link;

}bank;

bank \*h;

*void* create(){

*short* n,i;

    bank \*q;

    printf("Enter no of accounts : ");

    scanf("%hi",&n);

    printf("Enter account no, name, bank name,cur. balance of %hi accounts\n",n);

    for(i=0;i<n;i++){

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

        t->link=NULL;

        scanf("%hi%s%s%d",&t->no,t->name,t->bnm,&t->bal);

        if(h==NULL){

            h=t;

            q=t;

        }else{

            q->link=t;

            q=t;

        }

    }

}

*void* withdraw(){

*short* ano;

    bank \*q;

    q=h;

    printf("Enter account no to with draw : ");

    scanf("%hi",&ano);

    while(1){

        if(ano==q->no){

            printf("Enter amount to withdtaw : ");

*int* amo;

            scanf("%d",&amo);

            if(amo<=q->bal){

                printf("Withdraw successfull!\n");

                q->bal-=amo;

            }else{

                printf("Insufficent balance!!\n");

            }

            break;

        }

        if(q->link==NULL){

            printf("Invalid bank account\n");

            break;

        }

        q=q->link;

    }

}

*void* deposit(){

*short* ano;

    bank \*q;

    q=h;

    printf("Enter account no to deposite : ");

    scanf("%hi",&ano);

    while(1){

        if(ano==q->no){

            printf("Enter amount to deposit : ");

*int* amo;

            scanf("%d",&amo);

            if(amo<=q->bal){

                printf("Deposite successfull!\n");

                q->bal+=amo;

            }

            break;

        }

        if(q->link==NULL){

            printf("Invalid bank account\n");

            break;

        }

        q=q->link;

    }

}

*void* removal(){

*short* ano;

    bank \*q;

    printf("Enter account no to remove : ");

    scanf("%hi",&ano);

    if(h==NULL){//zero acc's

        printf("No accounts to remove\n");

        return;

    }

    else if(h->link==NULL){//only one acc's

        if(h->no==ano){

            free(h);

            printf("Removed details of acc no %hi \n",ano);

            h=NULL;

        }else{

            printf("Invalid acc no\n");

        }

        return;

    }

    q=h;

    while(1){

        if(h->no==ano){//checking 1st acc

            q=h->link;

            free(h);

            printf("Removed details of acc no %hi \n",ano);

            h=q;

            return;

        }else if(q->link->no==ano){

            bank \*t;

            t=q->link;

            printf("Removed details of acc no %hi \n",ano);

            q->link=q->link->link;

            free(t);

            return;

        }

        if(q->link==NULL){

            printf("Invalid bank account\n");

            break;

        }

        q=q->link;

    }

}

*void* display(){

    bank \*q;

    q=h;

    if(h==NULL){

        printf("No accounts\n");

        return;

    }

    while(q->link!=NULL){

        printf("%hi\t%s\t%s\t%d\n",q->no,q->name,q->bnm,q->bal);

        q=q->link;

    }

    printf("%hi\t%s\t%s\t%d\n",q->no,q->name,q->bnm,q->bal);

}

*void* main(){

    h=NULL;

*short* choice,repeat;

    create();

    do{

        printf("Enter choice to perform 1 withdraw, 2 deposite, 3 remove, 4 display, # exit : ");

        scanf("%hi",&choice);

        switch(choice){

            case 1:withdraw();

                   break;

            case 2:deposit();

                   break;

            case 3:removal();

                   break;

            case 4:display();

                   break;

            default:exit(1);

        }

        printf("repeat - #\t end - 0 :");

        scanf("%hi",&repeat);

    }while(repeat);

}

