NAME:VISHAL SINGH.

ROLL:12939.

CLASS:SYBCA(SCI).

//DATA\_STRUCTURES(2020-2021)//

PRACTICAL ASSIGNMENTS:-

1) Write a C program to accept and sort n elements in ascending order by using bubble sort. Non-recursive.

Code:- //Write a C program to accept and sort n elements in ascending order by using bubble sort using function.//

#include<stdio.h>

void read\_array(int[],int);

void display\_array(int[],int n);

void bubble\_sort(int[],int n);

int main()

{

    int arr[10];

    int n;

    printf("enter how many numbers u want\n");

    scanf("%d",&n);

    read\_array(arr,n);

     printf("before sorting array elements\n");

     printf("number are as follows\n");

    display\_array(arr,n);

    bubble\_sort(arr,n);

     printf("\nafter sorting the array element\n");

      printf("number are as follows\n");

    display\_array(arr,n);

    return 1;

}

 void read\_array(int arr[],int n)

    {

        int i;

      for(i=0;i<n;i++)

      {

        printf("arr[%d]=",i);

        scanf("%d",&arr[i]);

      }

    }

    void display\_array(int arr[],int n)

    {

        int i;

        for(i=0;i<n;i++)

        {

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

        }

    }

   void bubble\_sort(int arr[],int n)

   {

       int i,j,t,flag=-1,cmpcnt=-1,swapcnt=-1;

      for(i=1;i<n;i++)

       {

        flag=0; //assume no swapping

        for(j=0;j<n-i;j++)

        {

            cmpcnt++;

            if(arr[j]>arr[j+1])

            {

                swapcnt++;

                t=arr[j];

                arr[j]=arr[j+1];

                arr[j+1]=t;

                flag=1; //swapping happen

            }

        }

        if(flag==0)

         break;

    }

   printf("total compare=%d\n",cmpcnt);

   printf("total swapping=%d\n",swapcnt);

}

2) Write a C program to accept and sort n elements in ascending order by using insertion sort. Non-recursive.

Code:- #include<stdio.h>

void read\_array();

void display\_array();

void insertion\_sort();

int main()

{

    int arr[10];

    int n;

     printf("enter how many numbers u want\n");

    scanf("%d",&n);

    read\_array(arr,n);

     printf("\nbefore sorting the array element\n");

     display\_array(arr,n);

     insertion\_sort(arr,n);

      printf("\nafter sorting the array element\n");

      display\_array(arr,n);

}

void read\_array(int arr[10],int n)

{

    int i;

   for(i=0;i<n;i++)

    {

        printf("arr[%d]=",i);

        scanf("%d",&arr[i]);

    }

}

void display\_array(int arr[10],int n)

{

    int i;

     for(i=0;i<n;i++)

    {

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

    }

}

void insertion\_sort(int arr[10],int n)

{

    int i,j,key;

for(i=1;i<n;i++)

    {

        key=arr[i];

        for(j=i-1;(j>=0) && (key<arr[j]);j--)

        {

            if(arr[j]>key)

             arr[j+1]=arr[j];

             else

             break;

        }

        arr[j+1]=key;

    }

    // printf("total compare=%d\n",cmpcnt);

}

3) Write a program in C to accept 5 numbers from the user and sort the numbers in ascending order by using Merge sort. Recursive.

Code:- #include<stdio.h>

#include<stdlib.h>

void merge\_sort();

void merge();

void merge\_sort(int arr[],int low,int high)

{

   int mid;

   if(low<high)

   {

       mid=(low+high)/2;

       merge\_sort(arr,low,mid);

       merge\_sort(arr,mid+1,high);

       merge(arr,low,mid,high);

   }

}

void merge(int arr[],int low,int mid,int high)

{

    int t[50],i,j,k;

    i=low;

    j=mid+1;

    k=low;

    while((i<=mid) && (j<=high))

    {

        if(arr[i]>=arr[j])

          t[k++]=arr[j++];

        else

         t[k++]=arr[i++];

    }

    while(i<=mid)

        t[k++]=arr[i++];

    while(j<=high)

    t[k++]=arr[j++];

    for(i=low;i<=high;i++)

     arr[i]=t[i];

}

int main()

{

   int i, n, \*arr;

   printf("How many elements are u going to enter?: ");

   scanf("%d",&n);

   arr=(int\*)malloc(n\*sizeof(int));

   printf("Enter %d elements: ", n);

   for(i=0;i<n;i++)

      scanf("%d",&arr[i]);

   merge\_sort(arr,0,n-1);

   printf("Order of Sorted elements: ");

   for(i=0;i<n;i++)

      printf(" %d",arr[i]);

   return 0;

}

4) Write a C program to sort a random array of n integers by using Merge Sort algorithm in ascending order.Recursive

Code:- #include<stdio.h>

#include<stdlib.h>

void merge\_sort();

void merge();

void generate\_random();

void merge\_sort(int arr[],int low,int high)

{

   int mid;

   if(low<high)

   {

       mid=(low+high)/2;

       merge\_sort(arr,low,mid);

       merge\_sort(arr,mid+1,high);

       merge(arr,low,mid,high);

   }

}

void merge(int arr[],int low,int mid,int high)

{

    int t[50],i,j,k;

    i=low;

    j=mid+1;

    k=low;

    while((i<=mid) && (j<=high))

    {

        if(arr[i]>=arr[j])

          t[k++]=arr[j++];

        else

         t[k++]=arr[i++];

    }

    while(i<=mid)

        t[k++]=arr[i++];

    while(j<=high)

    t[k++]=arr[j++];

    for(i=low;i<=high;i++)

     arr[i]=t[i];

}

void generate\_random(int a[10],int n)

{

   int i;

   for(i=0;i<n;i++)

   a[i]=rand()%100;

}

int main()

{

   int i, n,arr[10];

   printf("How many elements are u going to enter?: ");

   scanf("%d",&n);

    generate\_random(arr,n);

    printf("elements are as follows: ");

   for(i=0;i<n;i++)

      printf("%d\t",arr[i]);

   merge\_sort(arr,0,n-1);

   printf("Order of Sorted elements: ");

   for(i=0;i<n;i++)

      printf(" %d",arr[i]);

   return 0;

}

5) Write a C program to accept n elements from user store it in an array. Accept a value from the user and use Non- recursive binary search method to check whether the value is present in array or not. Display proper message in output.

Code:- #include<stdio.h>

int main()

{

    int A[100];

    int x,n,i;

    int flag=0;

    printf("enter how many number u want\n");

    scanf("%d",&n);

    printf("enter array content one by one\n");

        for(i=0;i<n;i++)

        {

            printf("A[%d]=",i);

            scanf("%d",&A[i]);

        }

    printf("enter the number to be search\n");

    scanf("%d",&x);

   int lower=0;

   int higher=n-1;

   int middle=(lower+higher)/2;

   while(lower<=higher)

   {

       if(A[middle]==x)

          {

              printf("%d is found at %d location\n",x,middle+1);

               flag=1;

               break;

          }

         else if(A[middle] < x)

           lower=middle+1;

           else

           higher=middle-1;

           middle=(lower+higher)/2;

   }

   if(flag==0)

   printf("%d is not found at any location\n",x);

}

6) Write a C program to accept n elements from user store it in an array. Accept a value from the user and use recursive binary search method to check whether the value is present in array or not.

Code:- /\*Write a C program to accept n elements from user store it in an array. Accept a value from the user

and use recursive binary search method to check whether the value is present in array or not.\*/

#include<stdio.h>

int binary\_search();

int main()

{

    int arr[100],n;

    int i;

    printf("enter how many number u want\n");

    scanf("%d",&n);

    printf("array content\n");

    for(i=0;i<n;i++)

    {

        printf("arr[%d]=",i);

        scanf("%d",&arr[i]);

    }

    int x;

    printf("enter a number to be search\n");

    scanf("%d",&x);

    int k=binary\_search(arr,0,n-1,x);

    if(k==-1)

        {

            printf(" sorry given number is not found at any location in the array\n");

        }

        else

        {

            printf("%d is found at %d location in the array\n",x,k+1);

        }

        return 0;

}

int binary\_search(int arr[100],int lower,int higher,int x)

{

    int middle;

    if(lower>higher)

       return -1;

     middle=(lower+higher)/2;

    if(arr[middle]==x)

      return middle;

      else if( x < arr[middle] )

         binary\_search(arr,lower,middle-1,x);

         else

             binary\_search(arr,middle+1,higher,x);

}

7) Write a C program to implement a Singly linked list with following operations create() , display(),

insert(),delete().

Code:- //singly linked list with combined operations//

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

static int total;

struct node \*create(struct node \*head,int n)

{

    int i;

    struct node \*newnode=NULL,\*last=NULL;

    for(i=1;i<=n;i++)

    {

        newnode=(struct node\*)malloc(sizeof(struct node));

        if(newnode==NULL)

          printf("allocation is failed\n");

        else

        {

            printf("enter the node data value in it\n");

            scanf("%d",&newnode->data);

            newnode->next=NULL;

            if(head==NULL)

             head=last=newnode;

             else

             {

                 last->next=newnode;

                 last=newnode;

             }

        }

    }

    total=n;

    return head;

}

void display(struct node\*head)

{

    struct node \*temp=head;

    int cnt=0;

    printf("content of linked list are as follows\n");

      while(temp!=NULL)

      {

          printf("%d\t",temp->data);

           temp=temp->next;

           cnt++;

      }

      printf("\ntotal number of nodes=%d\n",cnt);

}

struct node \*insert(struct node \*head,int pos)

{

    struct node \*newnode=NULL, \*temp=NULL;

    int i;

    newnode=(struct node \*)malloc(sizeof(struct node));

    if(newnode==NULL)

      printf("allocation is failed\n");

    else

    {

        printf("enter the node data value in it\n");

        scanf("%d",&newnode->data);

        newnode->next=NULL;

        if(pos==1)

        {

            newnode->next=head;

            head=newnode;

        }

        else

        {

            temp=head;

            for(i=1;i<pos-1;i++)

              temp=temp->next;

            newnode->next=temp->next;

            temp->next=newnode;

        }

        total++;

     }

        return head;

}

struct node \*delete(struct node \*head,int pos)

{

    struct node \*temp=NULL ,\*p=NULL;

    int i;

    temp=head;

    if(pos==1)

         head=head->next;

    else

    {

        p=head;

        for(i=1;i<pos-1;i++)

         p=p->next;

        temp=p->next;

        p->next=temp->next;

    }

    printf("deleted element is %d\n",temp->data);

    temp->next=NULL;

    free(temp);

    total--;

    return head;

}

int main()

{

    struct node \*head=NULL;

    int n,ch,pos;

    do

    {

        printf("enter how many nodes u want\n");

        scanf("%d",&n);

    }while(n<=0);

    head=create(head,n);

    do

    {

        printf("\n 1.insert \n 2.delete \n 3.display\n 4.exit\n");

        printf("enter your choice\n");

        scanf("%d",&ch);

         switch(ch)

          {

             case 1:printf("enter the choice u want to insert the singly node?\n");

                    scanf("%d",&pos);

                    if(pos>0 && pos<=total+1)

                      head=insert(head,pos);

                    else

                      printf("sry!!..wrong position u entered\n");

                    break;

             case 2:if(head==NULL)

                      printf("singly linked list is empty\n");

                    else

                     {

                        printf("enter the position to be delete\n");

                        scanf("%d",&pos);

                        if(pos>0 && pos<=total)

                          head=delete(head,pos);

                        else

                          printf("sry!!..wrong position u entered\n");

                    }

                    break;

             case 3:if(head==NULL)

                     printf("singly linked is empty\n");

                    else

                     display(head);

                     break;

             case 4: return 0;

        }

    }while(ch>0 && ch<=4);

}

8) Write a C program to implement a Singly Circular linked list with following operations create(), display(), search(),length().

Code:- /\* Write a C program to implement a Singly Circular linked list with following operations create(), display(), search(),length().\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

static int total;

struct node \*create(struct node \*head,int n)

{

    int i;

    struct node \*newnode=NULL,\*last=NULL;

    for(i=1;i<=n;i++)

    {

        newnode=(struct node\*)malloc(sizeof(struct node));

        if(newnode==NULL)

          printf("allocation is failed\n");

        else

        {

            printf("enter the node data value in it\n");

            scanf("%d",&newnode->data);

            newnode->next=NULL;

            if(head==NULL)

             head=last=newnode;

             else

             {

                 last->next=newnode;

                 last=newnode;

             }

             last->next=head;

        }

    }

    total=n;

    return head;

}

void display(struct node\*head)

{

    struct node \*temp=head;

    int cnt=0;

    printf("content of linked list are as follows\n");

      while(temp->next!=head)

      {

          printf("%d-->",temp->data);

           temp=temp->next;

           cnt++;

      }

      printf("%d",temp->data);

      printf("\ntotal number of nodes=%d\n",cnt+1); //this is responsible for length//

}

int search(struct node \*head,int x)

{

    int i;

    struct node \*temp=head;

  for(i=1;temp->next!=head;i++)

    {

        if(temp->data==x)

            return i;

            temp=temp->next;

    }

    if(temp->data==x)  //this is responsible for last serach node

      return i;

    return 0;

}

int main()

{

    struct node \*head=NULL;

    int n,ch,pos,m,x;

    do

    {

        printf("enter how many nodes u want\n");

        scanf("%d",&n);

    }while(n<=0);

    head=create(head,n);

    do

    {

        printf("\n 1.display \n 2.search \n 3.exit\n");

        printf("enter your choice\n");

        scanf("%d",&ch);

         switch(ch)

          {

             case 1:if(head==NULL)

                     printf("singly linked is empty\n");

                    else

                     display(head);

                     break;

             case 2:if(head==NULL)

                     printf("singly linked is empty\n");

                    else

                    {

                        printf("enter the number to be search\n");

                        scanf("%d",&x);

                     m=search(head,x);

                     if(m!=0)

                         printf("given number is present in %d location\n",m);

                     else

                         printf("not found\n");

                    }

                     break;

             case 3: return 0;

        }

    }while(ch>0 && ch<=3);

}

9) Write a C program to implement a Doubly linked list with following operations create() , display(),

insert(),delete().

Code:- /\*Write a C program to implement a Doubly linked list with following operations create() , display(),

insert(),delete().\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*next;

    struct node\*prev;

};

static int total;

struct node \*create(struct node \*head,int n)

{

    int i;

    struct node\*temp;

    struct node\*newnode;

    newnode=temp=NULL;

    for(i=1;i<=n;i++)

    {

    newnode=(struct node\*)malloc(sizeof(struct node));

    printf("enter the data part\n");

    scanf("%d",&newnode->data);

    newnode->next=newnode->prev=NULL;

    if(head==NULL)

        head=temp=newnode;

    else

      {

        temp->next=newnode;

        newnode->prev=temp;

        temp=newnode;

      }

    }

    total=n;

    return head;

}

struct node \*insert(struct node \*head,int pos)

{

    int i;

    struct node\*temp;

    struct node\*newnode;

    newnode=(struct node\*)malloc(sizeof(struct node));

    printf("enter the data value\n");

    scanf("%d",&newnode->data);

    newnode->prev=newnode->next=NULL;

    if(pos==1)

    {

        if(head==NULL)

            head=newnode;

            else

            {

                newnode->next=head;

                head->prev=newnode;

                head=newnode;

            }

    }

    else

    {

        temp=head;

        for(i=1;i<pos-1;i++)

            temp=temp->next;

            newnode->next=temp->next;

            newnode->prev=temp;

            if(newnode->next!=NULL)

              (newnode->next)->prev=newnode;

            temp->next=newnode;

    }

    total++;

    return head;

}

struct node \*delete(struct node\*head,int pos)

{

    int i;

    struct node\*temp,\*q;

    temp=q=NULL;

    temp=head;

    if(pos==1)

    {

        if(head->next==head)

        {

            printf("deleted element:%d\n",head->data);

            free(head);

            total--;

            return head;

        }

        else

        {

            head=head->next;

            head->prev=NULL;

        }

    }

    else

    {

        q=head;

        for(i=1;i<pos-1;i++)

         q=q->next;

         temp=q->next;

         if(temp->next!=NULL)

           (temp->next)->prev=q;

        q->next=temp->next;

    }

    printf("deleted element:%d\n",temp->data);

    temp->prev=temp->next=NULL;

    free(temp);

    total--;

    return head;

}

void dispaly(struct node\*head)

{

    struct node\*temp=head;

    printf("content of linked list are as follows\n");

    while(temp!=NULL)

    {

        printf("%d\t",temp->data);

        temp=temp->next;

    }

}

int count(struct node\*head)

{

    int cnt=0;

    struct node\*temp=head;

    while(temp!=NULL)

    {

        cnt++;

        temp=temp->next;

    }

    return cnt;

}

int main()

{

    struct node\*head;

    head=NULL;

    int n,ch;

    int k;

    int pos;

    do

    {

        printf("enter how many nodes u want\n");

        scanf("%d",&n);

    }while(n<=0);

    head=create(head,n);

    do

    {

        printf("\n1:insertion\n2:deletion\n3:display\n4:count\n5:exit\n");

        printf("Enter your choice\n");

        scanf("%d",&ch);

        switch(ch)

        {

            case 1:printf("Enter the position to be inserted\n");

                    scanf("%d",&pos);

                        if(pos>0 && pos<=total+1)

                        head=insert(head,pos);

                    else

                    {

                    printf("invalid position");

                    }

                    break;

            case 2: if(head==NULL)

                        printf("linked list is empty\n");

                    else

                    {

                        printf("enter the position to be deleted\n");

                        scanf("%d",&pos);

                         if(pos>0 && pos<=total)

                        head=delete(head,pos);

                        else

                        {

                            printf("invalid position\n");

                        }

                        break;

                     }

            case 3:if(head==NULL)

                    printf("linked list is empty\n");

                    else

                        dispaly(head);

                    break;

            case 4:k=count(head);

                        printf("total nodes is %d\n",k);

                        break;

            case 5:return 0;

        }

    }while(ch>0 &&ch<6);

}

10) Write a C program to implement a Doubly Circular linked list with following operations create() and display(), append(),delete().

Code:- //circular doubly linked list using individual operation//

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*next;

    struct node\*prev;

};

static int total;

struct node \*last;

struct node \*create(struct node \*head,int n)

{

    int i;

    struct node \*newnode=NULL;

    for(i=1;i<=n;i++)

    {

    newnode=(struct node\*)malloc(sizeof(struct node));

    if(newnode==NULL)

     printf("allocation failed\n");

    printf("enter the data part\n");

    scanf("%d",&newnode->data);

    newnode->next=newnode->prev=NULL;

    if(head==NULL)

        head=last=newnode;

    else

      {

        last->next=newnode;

        newnode->prev=last;

        last=newnode;

        last->next=head;

      }

    }

    total=n;

    return head;

}

struct node \*insert\_beg(struct node \*head)

{

    struct node \*newnode=NULL;

     newnode=(struct node\*)malloc(sizeof(struct node));

     if(newnode==NULL)

     printf("allocation failed\n");

    printf("enter the data part\n");

    scanf("%d",&newnode->data);

    newnode->next=newnode->prev=NULL;

    newnode->next=head;

    newnode->prev=head->prev;

    head->prev=newnode;

    head=newnode;

    last->next=head;

     total++;

     return head;

}

struct node \*insert\_end(struct node \*head)

{

    struct node \*newnode=NULL;

     newnode=(struct node\*)malloc(sizeof(struct node));

     if(newnode==NULL)

     printf("allocation failed\n");

    printf("enter the data part\n");

    scanf("%d",&newnode->data);

    newnode->next=newnode->prev=NULL;

    newnode->next=last->next;

    newnode->prev=last;

    last->next=newnode;

    last=newnode;

    head->prev=newnode;

    total++;

    return head;

}

struct node \*insert\_middle(struct node \*head,int pos)

{

    struct node \*newnode=NULL ,\*temp=NULL;

    int i;

    temp=head;

     newnode=(struct node\*)malloc(sizeof(struct node));

     if(newnode==NULL)

     printf("allocation failed\n");

    printf("enter the data part\n");

    scanf("%d",&newnode->data);

    newnode->next=newnode->prev=NULL;

     for(i=1;i<pos-1;i++)

       temp=temp->next;

       newnode->prev=temp;

       newnode->next=temp->next;

       (temp->next)->prev=newnode;

       temp->next=newnode;

     printf("%d\n",head->prev->data);

      printf("%d\n",last->next->data);

         total++;

         return head;

}

struct node \*delete\_beg(struct node \*head)

{

    struct node \*temp=NULL;

    temp=head;

    printf("delete from front:%d\n",temp->data);

    head=head->next;

    head->prev=temp->prev;

    last->next=head;

    temp->next=temp->prev=NULL;

    free(temp);

    total--;

    return head;

}

struct node \*delete\_end(struct node \*head)

{

    struct node \*temp=NULL;

    temp=head;

    while((temp->next)->next!=head)

       temp=temp->next;

    last=temp;

    temp=temp->next;

    printf("deleted element:%d\n",temp->data);

    last->next=head;

    head->prev=last;

    temp->next=temp->prev=NULL;

    free(temp);

    total--;

    return head;

}

struct node \*delete\_middle(struct node \*head,int pos)

{

    struct node \*temp=NULL;

    struct node \*q;

    temp=head;

    int i;

    if(pos==1)

    {

        if(head->next==last->prev)

        {

            printf("deleted element is %d\n",temp->data);

            head=temp=last=NULL;

            free(temp);

            free(head);

            free(last);

        }

        else

        {

             head=head->next;

             head->prev=temp->prev;

             last->next=head;

             temp->next=temp->prev=NULL;

             free(temp);

        }

    }

    else

    {

       for(i=1;i<pos-1;i++)

           temp=temp->next;

        q=temp->next;

        temp->next=q->next;

        (q->next)->prev=temp;

        q->next=q->prev=NULL;

        free(q);

    }

     total--;

    return head;

}

void display(struct node \*head)

{

    struct node \*temp=head;

    while(temp->next!=head)

    {

        printf("%d\t",temp->data);

        temp=temp->next;

    }

    printf("%d\n",temp->data);

 }

int main()

 {

     struct node \*head=NULL;

    int n,ch,pos;

    do

    {

        printf("enter how many nodes u want\n");

        scanf("%d",&n);

    }while(n<=0);

    head=create(head,n);

    do

    {

        printf("\n 1.insert at beginning\n 2.insert at end \n 3.insert at middle\n 4.delete from beg\n 5.delete from end\n 6.delete from middle\n 7.display\n 8.exit\n");

        printf("enter your choice\n");

        scanf("%d",&ch);

         switch(ch)

          {

             case 1:head=insert\_beg(head);

                    break;

             case 2:head=insert\_end(head);

                     break;

            case 3:printf("enter the position to be inserted\n");

                   scanf("%d",&pos);

                   head=insert\_middle(head,pos);

                      break;

            case 4: head=delete\_beg(head);

                      break;

             case 5: head=delete\_end(head);

                      break;

             case 6:printf("enter the position to be deleted\n");

                   scanf("%d",&pos);

                    head=delete\_middle(head,pos);

                      break;

             case 7:display(head);

                     break;

             case 8: return 0;

        }

    }while(ch>0 && ch<=8);

 }

11) Write a C program to implement Static implementation of stack of integers with following operation:

a) Initialize()

b) push()

c) pop()

d) isempty()

e) isfull()

f) display()

g) peek()

code:-                                   //stack implementation using static(ARRAY) and isempty and isfull with structure //

#include<stdio.h>

#include "1.h" //here i include my own header file

int isfull();

int isempty();

void push();

void pop();

void peek();

void display();

int main()

{

    int ch;

    vishal s1;

    init(&s1);

    do

     {

      printf("\n1:push\n2:pop\n3:peek\n4:display\n5:exit\n");

      printf("enter the choice\n");

      scanf("%d",&ch);

        switch(ch)

         {

            case 1:if(isfull(&s1))

                    printf("\nstack is overflow\n");

                   else

                    push(&s1);

                   break;

            case 2:if(isempty(&s1))

                    printf("\nstack is underflow\n");

                   else

                    pop(&s1);

                   break;

            case 3:peek(&s1);

                   break;

            case 4:display(&s1);

                   break;

            case 5:return 0;

                   break;

            default:printf("\ninvalid choice\n");

        }

    } while (ch>0 && ch<=5);

}

Header file------

#include<stdio.h>

#define n 5

typedef struct stack

{

    int data [n];

    int top;

}vishal;

void init(vishal \*s)

{

    s->top=-1;

}

int isfull(vishal \*s)

{

    if(s->top==n-1)

     return 1;

     else

     return 0;

}

int isempty(vishal \*s)

{

    if(s->top==-1)

      return 1;

    else

     return 0;

}

void push(vishal \*s)

{

    int x;

    printf("enter the data \n");

    scanf("%d",&x);

      s->data[++s->top]=x;

}

void pop(vishal \*s)

{

      s->data[s->top--];

}

void peek(vishal \*s)

{

     printf("peek=%d\n",s->data[s->top]);

}

void display(vishal \*s)

{

    int i;

    for(i=s->top;i>=0;i--)

      {

        printf("%d\t",s->data[i]);

      }

}

12) Write a C program to implement Dynamic implementation of stack of integers with following

operation:

h) Initialize()

i) push()

j) pop()

k) isempty()

l) isfull()

m) display()

n) peek()

code:-                                             //stack using dynamic memory allocation(linked-list) and isempty//

#include<stdio.h>

#include "3.h"

int main()

{

    int ch;

    do

    {

        printf("\n1:push\n2:pop\n3:peek\n4:display\n5:exit\n");

        printf("enter the choice\n");

        scanf("%d",&ch);

        switch(ch)

         {

            case 1:push();

                    break;

            case 2:if(isempty())

                    printf("\nstack is underflow\n");

                   else

                    pop();

                    break;

            case 3:peek();

                    break;

            case 4:display();

                    break;

            case 5:return 0;

                    break;

            default:printf("\ninvalid choice\n");

        }

    } while (ch>0 && ch<=5);

}

Header file----

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*next;

};

struct node\*top=0;

int isempty()

{

    if(top==0)

      return 1;

    else

     return 0;

}

void push()

{

    int x;

    printf("enter the data \n");

    scanf("%d",&x);

    struct node\*newnode;

     newnode=(struct node\*)malloc(sizeof(struct node));

     newnode->data=x;

     newnode->next=top;

     top=newnode;

}

void pop()

{

    struct node\*temp;

    temp=top;

    if(top==NULL)

     printf("underflow condition\n");

     else

     {

        printf("%d",top->data);

        top=top->next; //temp->next;

        free(temp);

     }

}

void peek()

{

   if(top==NULL)

    printf("underflow condition\n");

    else

     printf("%d\t",top->data);

}

void display()

{

   struct node\*temp=top;

    if(top==0)

      printf("underflow condition\n");

    else

    {

        while(temp!=NULL)

         {

             printf("%d\t",temp->data);

             temp=temp->next;

         }

    }

}

Q.Following programs are supposed to be solved or written by all.

1. Write a C program to create binary search tree of integers and perform following operations:

• Preordertraversal

• Post ordertraversal

Code:- //printf("\n1:create\n2:inorder\n3:insert\n4:postorder\n5:preorder\n6:searching\n7:exit\n");

#include <stdio.h>

#include <stdlib.h>

struct node

 {

  int key;

  struct node \*left, \*right;

};

// Creation of simple node

struct node \*newNode(int item)

 {

  struct node \*temp = (struct node \*)malloc(sizeof(struct node));

  temp->key = item;

  temp->left = temp->right = NULL;

  return temp;

}

void inorder(struct node \*root)

{

  if (root != NULL)

   {

     inorder(root->left);

     printf("%d -> ", root->key);

     inorder(root->right);

  }

}

void postorder(struct node \*root)

{

  if (root != NULL)

   {

     postorder(root->left);

     postorder(root->right);

     printf("%d -> ", root->key);

  }

}

void preorder(struct node \*root)

{

  if (root != NULL)

   {

     printf("%d -> ", root->key);

     preorder(root->left);

     preorder(root->right);

  }

}

struct node \*insert(struct node \*node, int key)

 {

  if (node == NULL)

    return newNode(key);

  if (key < node->key)

    node->left = insert(node->left, key);

  else

    node->right = insert(node->right, key);

  return node;

}

struct node \*create(struct node \*node, int key)

 {

  if (node == NULL)

    return newNode(key);

if (key < node->key)

    node->left = insert(node->left, key);

  else

    node->right = insert(node->right, key);

  return node;

}

struct node \*search(struct node \*tree,int x)

{

    if(tree!=NULL)

      {

          if(tree->key==x)

            return tree;

        if(x < tree->key)

         return(search(tree->left,x));

         else

          return(search(tree->right,x));

      }

}

int main()

{

  struct node \*tree = NULL,\*temp=NULL;

  int ch,x,n;

  do

  {

      printf("\n1:create\n2:inorder\n3:insert\n4:postorder\n5:preorder\n6:searching\n7:exit\n");

      printf("enter the choice\n");

      scanf("%d",&ch);

     switch(ch)

     {

         case 1:printf("enter the data\n");

                scanf("%d",&x);

                tree=create(tree,x);

                break;

         case 2:printf("Inorder traversal: ");

                inorder(tree);

                break;

         case 3:printf("enter the data\n");

                scanf("%d",&x);

                tree=insert(tree,x);

                break;

         case 4:printf("postorder traversal: ");

                postorder(tree);

                break;

         case 5:printf("preorder traversal: ");

                preorder(tree);

                break;

        case 6:if(tree==NULL)

                 printf("tree is NULL\n");

                else

                 {

                   printf("enter key node to be search in a tree\n");

                   scanf("%d",&x);

                   temp=search(tree,x);

                    if(temp!=NULL)

                     printf("%d node is found\n",temp->key);

                    else

                     printf("sry keynode is found\n");

                 }

                    break;

        case 7:return 0;

     }

 }while(ch>0 && ch<=7);

 }

2. Write a C program to read a graph as adjacency matrix and display the adjacency matrix.

Code:- /\*2. Write a C program to read a graph as adjacency matrix and display the adjacency matrix.\*/

#include <stdio.h>

#include <stdlib.h>

struct AdjListNode

{

int dest;

struct AdjListNode\* next;

};

// A structure to represent an adjacency list

struct AdjList

{

struct AdjListNode \*head;

};

struct Graph

{

int V;

struct AdjList\* array;

};

struct AdjListNode\* newAdjListNode(int dest)

{

struct AdjListNode\* newNode =

(struct AdjListNode\*) malloc(sizeof(struct AdjListNode));

newNode->dest = dest;

newNode->next = NULL;

return newNode;

}

struct Graph\* createGraph(int V)

{

struct Graph\* graph =

(struct Graph\*) malloc(sizeof(struct Graph));

graph->V = V;

graph->array =

(struct AdjList\*) malloc(V \* sizeof(struct AdjList));

int i;

for (i = 0; i < V; ++i)

graph->array[i].head = NULL;

return graph;

}

void addEdge(struct Graph\* graph, int src, int dest)

{

struct AdjListNode\* newNode = newAdjListNode(dest);

newNode->next = graph->array[src].head;

graph->array[src].head = newNode;

newNode = newAdjListNode(src);

newNode->next = graph->array[dest].head;

graph->array[dest].head = newNode;

}

void printGraph(struct Graph\* graph)

{

int v;

for (v = 0; v < graph->V; ++v)

{

struct AdjListNode\* pCrawl = graph->array[v].head;

printf("\n Adjacency list of vertex %d\n head ", v);

while (pCrawl)

{

printf("-> %d", pCrawl->dest);

pCrawl = pCrawl->next;

}

printf("\n");

}

}

void main()

{

int V = 5;

struct Graph\* graph = createGraph(V);

addEdge(graph, 0, 1);

addEdge(graph, 0, 4);

addEdge(graph, 1, 2);

addEdge(graph, 1, 3);

addEdge(graph, 1, 4);

addEdge(graph, 2, 3);

addEdge(graph, 3, 4);

printGraph(graph);

}

3. Add a function in Q2 (above question) to count total degree, indegree and outdegree of the graph.

Code:- #include <stdio.h>

#include <stdlib.h>

typedef struct node

{

    int data;

    struct node \*left, \*right;

} node;

void create(node \*\*r)

{

    int n, i;

    node \*cur, \*newnode, \*par;

    printf("How many nodes u wants to enter :");

    scanf("%d", &n);

    for (i = 0; i < n; i++)

    {

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

        scanf("%d", &newnode->data);

        if (\*r == NULL)

        {

            \*r = newnode;

            cur = newnode;

            cur->left = cur->right = NULL;

        }

        else

        {

            cur = \*r;

            while (cur != NULL)

            {

                par = cur;

                if (cur->data >= newnode->data)

                {

                    cur = cur->left;

                }

                else

                {

                    cur = cur->right;

                }

            }

            if (par->data >= newnode->data)

            {

                par->left = newnode;

            }

            else

            {

                par->right = newnode;

            }

        }

    }

}

void preorder(node \*r)

{

    if (r == NULL)

        return;

    printf("%d ", r->data);

    preorder(r->left);

    preorder(r->right);

}

void postorder(node \*r)

{

    if (r == NULL)

        return;

    preorder(r->left);

    preorder(r->right);

    printf("%d ", r->data);

}

void main(int argc, char \*argv[]) //command line argument//

{

    node \*root = NULL;

    printf("Create a BST : ");

    create(&root);

    printf("\nPREORDER : ");

    preorder(root);

    printf("\nPOSTORDER : ");

    postorder(root);

}