**1]ADAM FUNCTION:**

#include<stdio.h>

int reverse(int num)

{

int rem,r=0;

while(num>0)

{

rem=num%10;

r=r\*10+rem;

num=num/10;

}

return r;

}

int main()

{

int num,s1,r,s2,r1;

scanf("%d",&num);

s1=num\*num;

r=reverse(num);

s2=r\*r;

r1=reverse(s2);

if(s1==r1)

{

printf("adam num");

}

else

{

printf("not adam num");

}

return 0;

}

**2]ARRAY PROGRAMS:**

#include<stdio.h>

int main()

{

int n,i,pos,value;

printf("enter the size: ");

scanf("%d",&n);

int a[n+1];

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

printf("enter the pos: ");

scanf("%d",&pos);

printf("enter the value to be inserted: ");

scanf("%d",&value);

for(i=n-1;i>=pos;--i)

{

a[i+1]=a[i];

}

a[pos]=value;

for(i=0;i<=n;i++)

{

printf("%d",a[i]);

}

return 0;

}

3]#include<stdio.h>

int main()

{

int n,i,pos;

printf("enter the size: ");

scanf("%d",&n);

int a[n];

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

printf("enter the pos to be deleted: ");

scanf("%d",&pos);

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

{

a[i]=a[i+1];

}

for(i=0;i<n-1;i++)

{

printf("%d",a[i]);

}

return 0;

}

**4]Reverse without using built in function:**

#include<stdio.h>

#include<string.h>

int main ()

{

char a[10],b[10];

gets(a);

int k=0,i,j;

for(i=0;a[i]!='\0';++i);

for(j=i-1;j>=0;--j)

{

b[k]=a[j];

++k;

}

b[k]='\0';

printf("%s",b);

return 0;

}

**5]LINKED LIST:**

#include<stdio.h>

struct node

{

int data;

struct node\*next;

};

int choice=1,ele;

struct node \*newnode,\*start=0,\*end=0;

void create()

{

while (choice==1)

{

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

printf("enter element:\n");

scanf("%d",&ele);

newnode->data=ele;

newnode->next=0;

if(start==0)

{

start=newnode;

end=newnode;

}

else

{

end->next=newnode;

end=newnode;

}

end->next=start;

printf("do you want to continue");

scanf("%d",&choice);

}

}

void display()

{

struct node\*temp=start;

while(temp->next!=start)

{

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

temp=temp->next;

}

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

}

int main()

{

create();

display();

return 0;

}

**6]CYCLIC SORT:**

#include<stdio.h>

void swap(int a[],int first,int second)

{

int temp=a[first];

a[first]=a[second];

a[second]=temp;

}

void cyclicsort(int a[],int n)

{

int i=0,correct;

while(i<n)

{

correct=a[i]-1;

if(a[i]!=a[correct])

{

swap(a,i,correct);

}

else

{

++i;

}

}

}

int main()

{

int n,i;

scanf("%d",&n);

int a[n];

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

cyclicsort(a,n);

for(i=0;i<n;i++)

{

printf("%d",a[i]);

}

return 0;

}

**7]DOUBLY LINKED LIST:**

#include<stdio.h>

struct node

{

int data;

struct node\*next;

struct node\*pre;

};

int choice=1,ele;

struct node \*newnode,\*start=0,\*end=0;

void create()

{

while (choice==1)

{

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

printf("enter element:\n");

scanf("%d",&ele);

newnode->data=ele;

newnode->next=0;

newnode->pre=0;

if(start==0)

{

start=newnode;

end=newnode;

}

else

{

end->next=newnode;

newnode->pre=end;

end=newnode;

}

end->next=start;

start->pre=end;

printf("do you want to continue");

scanf("%d",&choice);

}

}

void display()

{

struct node\*temp=start;

while(temp->next!=start)

{

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

temp=temp->next;

}

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

}

int main()

{

create();

display();

return 0;

}

**8]DOUBLY LINKED LIST:**

#include<stdio.h>

struct node

{

int data;

struct node\*next;

struct node\*pre;

};

int choice=1,ele;

struct node \*newnode,\*start=0,\*end=0;

void create()

{

while (choice==1)

{

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

printf("enter element:\n");

scanf("%d",&ele);

newnode->data=ele;

newnode->next=0;

newnode->pre=0;

if(start==0)

{

start=newnode;

end=newnode;

}

else

{

end->next=newnode;

newnode->pre=end;

end=newnode;

}

printf("do you want to continue");

scanf("%d",&choice);

}

}

void display()

{

struct node\*temp=start;

while(temp!=0)

{

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

temp=temp->next;

}

}

int main()

{

create();

display();

return 0;

}

**9]LEETCODE-ORDER AGNOSTIC BINARY SEARCH:**

#include <stdio.h>

#include <stdbool.h>

int peak(int a[], int n){

int start=0,end=n-1,mid;

while(start<end)

{

mid=(start+end)/2;

if(a[mid]>a[mid+1] )

{

end=mid;

}

else

{

start=mid+1;

}

}

return start;

}

int oabs(int a[],int n,int target,int res)

{

int start=0;

int end=n-1;

bool Asc=a[res]>target;

while(start<=end)

{

if(a[res]==target)

{

return res;

}

if(Asc)

{

if(target<a[res])

{

end=res-1;

}

else

{

start=res+1;

}

}

else

{

if(target>a[res])

{

start=res+1;

}

else

{

end=res-1;

}

}

}

return -1;

}

int main()

{

int n,i;

printf("enter n:\n");

scanf("%d",&n);

int a[n];

printf("enter array values:\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

int target;

printf("enter target:\n");

scanf("%d",&target);

int res=peak(a,n);

int R=oabs(a,n,target,res);

printf("%d",R);

return 0;

}

**10]LEETCODE:**

#include<stdio.h>

#include<stdbool.h>

int searchRange(int nums[], int numsSize, int target)

{

int start=0,end=numsSize-1,mid;

int returnSize[2];

while(start<=end)

{

mid=(start+end)/2;

if(target>nums[mid])

{

start=mid+1;

}

else if(target<nums[mid])

{

end=mid-1;

}

else

{

return returnSize[mid,mid-1];

}

}

return returnSize[-1,-1];

}

int main()

{

int nums[20],i,target,numsSize;

printf("enter n:\n");

scanf("%d",&numsSize);

for(i=0; i<numsSize; i++)

{

scanf("%d",&nums[i]);

}

printf("enter target:\n");

scanf("%d",&target);

int R=searchRange(nums,numsSize,target);

printf("%d",R);

}

**11]MACROS:**

#include<stdio.h>

#define MUL(a,b) a\*b

int main()

{

printf("%d",MUL(5-2,7+4));

return 0;

}

**12]LEETCODE-OUTER INCREASING PATTERN:**

#include<stdio.h>

int min(int a,int b)

{

if(a<b)

{

return a;

}

else

{

return b;

}

}

int main()

{

int i,j,n,N,left,right,top,bottom,res;

scanf("%d",&n);

N=2\*n;

for(i=0;i<=N;++i)

{

for(j=0;j<=N;++j)

{

left=j;

right=N-j;

top=i;

bottom=N-i;

res=n-(min(min(left,right),min(top,bottom)));

printf("%d",res);

}

printf("\n");

}

return 0;

}

**11]STRONG NUM:**

#include<stdio.h>

int main()

{

int n,rem,i,fact,sum=0,temp;

scanf("%d",&n);

temp=n;

if(n==0)

{

printf("not a strong");

return 0;

}

while(n>0)

{

rem=n%10;

fact=1;

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

{

fact \*=i;

}

sum+=fact;

n=n/10;

}

printf("%d\n",sum);

if(sum==temp)

{

printf("it is strong num");

}

else

{

printf("it is not strong num");

}

return 0;

}

**12]NUM PATTERN**:

#include<stdio.h>

#include<math.h>

int main()

{

int n,s,c=0,d,sum,flag=0,temp,i;

scanf("%d",&n);

s=n\*n;

temp=s;

while(s>0)

{

s=s/10;

++c;

}

s=temp;

for(i=1;i<=c-1;++i)

{

d=pow(10,i);

if(d==n)

{

continue;

}

sum=s/d+s%d;

if(sum==n)

{

flag=1;

break;

}

}

if(flag==1)

{

printf("kaprekar");

}

else

{

printf("not kaprekar");

}

return 0;

}

**13]ZOHO PATTERN:**

#include<stdio.h>

int min(int a,int b)

{

if(a<b)

{

return a;

}

else

{

return b;

}

}

int main()

{

int i,j,n,N,left,right,top,bottom,res;

scanf("%d",&n);

N=2\*n;

for(i=0;i<=N;++i)

{

for(j=0;j<=N;++j)

{

left=j;

right=N-j;

top=i;

bottom=N-i;

res=min(min(left,right),min(top,bottom));

printf("%d",res);

}

printf("\n");

}

return 0;

**}**

**14]QUEUE LINKED LIST:**

#include<stdio.h>

struct node

{

int data;

struct node\*next;

};

struct node\*newnode,\*front=0,\*rear=0;

void enqueue(int x)

{

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

newnode->data=x;

newnode->next=0;

if(front==0)

{

front=newnode;

rear=newnode;

}

else{

rear->next=newnode;

rear=newnode;

}

}

void dequeue()

{

struct node\*temp=front;

if(rear==0 || front==0)

{

printf("underflow");

}

else{

front=front->next;

free(temp);

}

}

void peek()

{

if(rear==0 || front==0)

{

printf("cant find peek");

}

else

{

printf("%d",front->data);

}

}

void display()

{

if(rear==0 || front==0)

{

printf("cant display");

}

else

{

struct node \*temp=front;

while(temp!=0)

{

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

temp=temp->next;

}

}

}

int main()

{

enqueue(10);

enqueue(20);

enqueue(30);

enqueue(40);

dequeue();

peek();

display();

return 0;

}