Day-1 Structures

```c
//Prime numbers from 1 to N
#include<stdio.h>
#include<stdbool.h>
bool isPrime(int n){
    if(n==1 || n==0)
        return false;
    for(int i=2;i<=n/2;i++){//In case of n==2 condition will be failed so it will return true
        /*
          Checking whether there are any factors below n/2
          If there are any then it is not a prime number.
          For Ex:let us check 10 is prime are or.
          From 2 to 5, 5 is a factor of 10.soo it is not a
          prime soo returning false.
        */
        if(n%i==0)
            return false;
    }
    return true;//Returns true if there are no factors since it will be a prime
}
int main() {
    int num;
    scanf("%d",&num);
    for(int i=1;i<=num;i++){
        if(isPrime(i))
            printf("%d\n",i);
    }
    return 0;
}
```

```c
// leap year
#include<stdio.h>
void main(){
    int year;
    printf("Enter a year:");
    scanf("%d",&year);
    if((year%4==0 && year%100!=0)||(year%400==0))
        printf("Leap year");
    else
        printf("Not a leap year");
}
```

```c
//Character pattern
#include<stdio.h>
void main(){
   int num,chare=65;
   scanf("%d",&num);
   for(int i=0;i<num;i++){
      for(int j=0;j<=i;j++)
         printf("%c ",chare++);
      printf("\n");
   }
}
/*
5
A
B C
D E F
G H I J
K L M N O
*/

//Reverse of an array
#include<stdio.h>
void main(){
   int size;
   printf("Enter size of array:");
   scanf("%d",&size);
   int arr[size];
   printf("Enter array elements:");
   for(int i=0;i<size;i++)
      scanf("%d",&arr[i]);
   printf("Reverse is..");
   for(int i=size-1;i>=0;i--)
      printf("%d",arr[i]);
}

//Rotate array by given value
#include<stdio.h>
void main(){
   int size,r;
   printf("Enter size of array:");
   scanf("%d",&size);
   printf("Enter rotation:");
```

```
    scanf("%d",&r);
    int arr[size];
    int farr[size];
    printf("Enter array elements:");
    for(int i=0;i<size;i++)
        scanf("%d",&arr[i]);
    int k=r,s=size;
    for(int i=0;i<size;i++){
        if(i<r){
        farr[size-k]=arr[i];
        k--;}
        else{farr[size-(s--)]=arr[i];}
    }
    for(int i=0;i<size;i++)
        printf("%d ",farr[i]);
}
/*
Enter size of array:5
Enter rotation:3
Enter array elements:1 5 4 3 2
3 2 1 5 4
*/
```

```
//Sample structure program
#include<stdio.h>
#include<string.h>
//creating struct with person1 variable
struct person{
    char name[50];
    int citno;
    float salary;
}person1;

int main()
{
    //assign value to name to person1
    strcpy(person1.name,"Geethanjali");
    //assign values to other person1 variables
    person1.citno =2014;
    person1.salary = 250000;
    printf("Name: %s \nCitno: %d\nsalary:
%.2f\n",person1.name,person1.citno,person1.salary);

    struct person person2;
```

```
    strcpy(person2.name,"D Pranai");
    person2.citno=2004;
    person2.salary=2000000;
    printf("Name: %s \nCitno: %d\nsalary:
%.2f\n",person2.name,person2.citno,person2.salary);

}
/*
Name: Geethanjali
Citno: 2014
salary: 250000.00
Name: D Pranai
Citno: 2004
salary: 2000000.00
*/

//Compare the struct syntax and declaration in main method of above and below
codes.Same same but different


#include<stdio.h>
#include<string.h>
//creating struct with person1 variable
struct {
    char name[50];
    int citno;
    float salary;
}person1,person2;
int main()
{

    //assign value to name to person1
    strcpy(person1.name,"Geethanjali");
    //assign values to other person1 variables
    person1.citno =2014;
    person1.salary = 250000;
    printf("Name: %s \nCitno: %d\nsalary:
%.2f\n",person1.name,person1.citno,person1.salary);

    strcpy(person2.name,"D Pranai");
    person2.citno=2004;
    person2.salary=2000000;
    printf("Name: %s \nCitno: %d\nsalary:
%.2f\n",person2.name,person2.citno,person2.salary);
```

```
}
/*
Name: Geethanjali
Citno: 2014
salary: 250000.00
Name: D Pranai
Citno: 2004
salary: 2000000.00
*/


//Stucture aliasing using typedef
#include<stdio.h>
typedef struct Distance{
   int feet;
   float inch;
} dist;
void main(){
   dist d1,d2;
}


//structure variables creation
//Method-1
struct Dist{
   float inch;
   int feet;
}d1,d2,d[20];//method-1 Either here or in main
struct Dist{
   int inch,feet;
};
void main(){
   struct Dist d1,d2,d[20];//method 2
}


/*
Accessing struct members
type-1: using . operator-member operator
type-2: using -> operator-structure pointer operator
*/
```

```c
//Nested structures
#include<stdio.h>
struct complex{
    int imag;
    float real;
};
struct number{
    struct complex comp;
    int integers;
}num1,num2;
//Assigning value to imag:
void main(){
num1.comp.imag=11;
num1.comp.real=12.6;
printf("%d\n %.2f",num1.comp.imag,num1.comp.real);
}

// either individual declaration and variable creation in one structure or Structure inside a structure
struct number{
    struct complex{
    int imag;
    float real;
    };
    int integers;
}num1,num2;


//Real life example of Nested structures
#include<stdio.h>
#include<string.h>
struct address{
    int doorno;
    int pincode;
    char address[20];
};
struct patient{
    struct address adr;
    int pid;
    int bill;
}p1,p2;
void main(){
```

```
p1.adr.doorno=4250;
p1.adr.pincode=524003;
strcpy(p1.adr.address,"Ramalingapuram");
p1.pid=1001;
p1.bill=15000;
printf("Patient
address:\nDoorno:%d\npincode:%d\naddress:%s\n",p1.adr.doorno,p1.adr.pincode,
p1.adr.address);
printf("Patient id:%d\nBill:%d\n",p1.pid,p1.bill);
}

// Like above using dot operator individual assignment or Like below using flower
brackets that follow the order of members of struct
#include<stdio.h>
#include<string.h>
struct details{
   int age;
   char name[30];
   struct dob{
      int day;
      int month;
      int year;
   }db;
   float salary;
};
void main(){
   struct details dt={25,"Pranai",{11,2,2004},1200000.52};
   // or
scanf("%d%s%d%d%d%f",&dt.age,&dt.name,&dt.db.day,&dt.db.month,&dt.db.year,&d
t.salary);
   printf("Age:%d\nName:%s\nDOB-%d %d
%d\nSalary:%.2f\n",dt.age,dt.name,dt.db.day,dt.db.month,dt.db.year,dt.salary);
}
```

Day-2 Strings

```
//String declaration
#include<stdio.h>
#include<string.h>
void main(){
   char ch[6]={'h','e','l','l','o','\0'};
```

```
    char st[]="Hello all";
    printf("Char array: %s\nString Literal: %s",ch,st);
}

//Traversing strings
//Method-1:Using length of the string
//Counting no of vowels and consonents in a string
#include<stdio.h>
#include<string.h>
void main(){
    char s[5]="Oreos";//Need to specify length to traverse using length
    int i=0,vc=0,cc=0;
    while(i<5){

if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u'||s[i]=='A'||s[i]=='E'||s[i]=='I'||s[i]=='O'||s[i]=='U')
            vc++;
        else
            cc++;
        i++;
    }
    printf("No of vowels:%d\n",vc);
    printf("No of consonents:%d",cc);
}
//No of vowels:3
//No of consonents:2

//Method-2:Using Null character
#include<stdio.h>
#include<string.h>
void main(){
    char s[]="Oreos";//No need to specify length in case of traversing using null
    int i=0,vc=0,cc=0;
    while(s[i]!=NULL){

if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u'||s[i]=='A'||s[i]=='E'||s[i]=='I'||s[i]=='O'||s[i]=='U')
            vc++;
        else
            cc++;
        i++;
    }
    printf("No of vowels:%d\n",vc);
    printf("No of consonents:%d",cc);
}
```

```
//No of vowels:3
//No of consonents:2

//Dynamic string allocation
#include<stdio.h>
#include<string.h>
void main(){
   char s[20];
   printf("Enter a string:");
   scanf("%s",s);//no need of & here in case of string
   printf("You entered %s",s);//it will terminate when it encountered space soo rest will be left alone
   //to overcome this we can give explicit instructions to compiler in the below way
   scanf("%[^\n]s",s);//It will read until it encounter nextline character//Not an error it is correct
}

//pointers with strings
#include<stdio.h>
#include<string.h>
void main(){
   char s[11]="Javapoint";
   char *p=s;
   printf("%s",p);
}
//Javapoint

#include<stdio.h>
#include<string.h>
void main(){
   char *p="Hello everyone";
   printf("String p:%s\n",p);
   char *q;
   printf("Copying the content of p to q..\n");
   q=p;
   printf("String q:%s\n",q);
}

//gets and puts and fgets
#include<stdio.h>
#include<string.h>
void main(){
   char s[20],h[20];
```

```
    printf("Enter a string1:");
    gets(s);
    printf("Enter a string2:");
    fgets(h,5,stdin);//can give bounds to the input text
    printf("value of string1:");
    puts(s);//To print
    printf("value of string2:");
    puts(h);
}
//With pointers
#include <stdio.h>
int main() {
    char a[11];
    char *b=a;
    fgets(b,5,stdin);
    puts(b);
    return 0;
}
```

```
//String functions
#include<stdio.h>
#include<string.h>
void main(){
    char ch1[7]="Javalk",ch2[7];
    printf("Length is %d\n\n",strlen(ch1));
    strcpy(ch2,ch1);
    printf("Value of second string is:%s\n\n",ch2);
    strcat(ch1,ch2);
    printf("Value of concatenated string is:%s\n\n",ch1);
    printf("Reversed string is %s\n\n",strrev(ch1));
    printf("Same or Different: %s\n\n",strcmp(ch1,ch2)?"Different":"Same");
    printf("Lower string is: %s\n\n",strlwr(ch2));
    printf("upper string is: %s\n\n",strupr(ch2));
}
//Check Password validity
#include<stdio.h>
#include<string.h>
int CheckPassword(char str[],int n){
    int ncount=0,lcount=0,ccount=0;
    for(int i=0;i<n;i++){
        if(str[i]==' ' || str[i]=='/' || (str[0]>47 && str[0]<58))
            return 0;
        if(str[i]>47 && str[i]<58)
```

```
            ncount++;
        if(str[i]>64 && str[i]<91 || str[i]>96 && str[i]<123)
            lcount++;
        if(str[i]>64 && str[i]<91)
            ccount++;
    }
    if(lcount>3 && ncount>0 && ccount>0)
        return 1;
    else
        return 0;
}
void main(){
    int size;
    printf("Enter size of password:");
    scanf("%d",&size);
    printf("Enter password");
    char str[size];
    scanf("%s",str);
    puts(str);
    if(!CheckPassword(str,size))
        printf("Not a valid password");
    else
        printf("Valid Password");
}

//Palindrome or not
#include<stdio.h>
#include<string.h>
void main(){
    char strs[5]="mamy";
    char dummy[5];
    strcpy(dummy,strs);
    if(dummy==strrev(strs))
        printf("Palindrome");
    else
        printf("Not a palindrome");
    return 0;
}
```

Day-3 Pointers

```
/*Pointers
Pointers always stores masked address but every variable will have both mask and original addrs
&-creates a mask or reference address to a variable.
Pointers has the capacity to store both values and address
&-Referencing operator
*-De Referencing operator
We can increment the address refered by a pointer For example: To store elements in an array.Refer line 133
*/









//%u-value %u-masked address %u-Next new masked address
//%p-hexadecimal value %p-original address %p-next new original address
#include<stdio.h>
#include<conio.h>
void main(){
   int b=20,*a=30;
printf("Value of b %p : %p -- %p\n",b);//Value of b 00000014 : 0061FF54 -- 00401A0B
printf("Value of a %p : %p -- %p\n",a);//Value of a 0000001E : 0061FF54 -- 00401A0B
printf("Value of b %u\n",b);//Value of b 20
printf("Value of a %u : %u--%u--%p--%p--%u--%p\n",a);//Value of a 30 : 6422356--4200971--004019B0--00741588--30--00000014
printf("Value of b: %d\n",b);//Value of b: 20
printf("Value of a : %d\n",a);//Value of a : 30
}

//Double pointer
#include<stdio.h>
void main(){
   int **a,b=10,c=30;
   a=&b;
   b=&c;
   printf("The output is:%d %d",*a,**a);//The output is:6422292(masked address) 30(value)
}
```

```c
//Null pointer
#include<stdio.h>
void main(){
    int *n=NULL;
    int m=200;
    printf("the output is:%d",*n+m);//200
}

//Void pointer
#include<stdio.h>
void main(){
    void *n=10;
    printf("the output is:%d",*n);//Not possible so raises error
}

//Wild pointer//Wild means cannot be changed
#include<stdio.h>
void main(){
    const int *a=10,b=20;
    a=&b;
    printf("%d",(*a+b));
    *a=90;//Error since pointer is a constant/wild pointer
    printf("%d",*a);
}

//Kill pointer
#include<stdio.h>
void main(){
    const int *a=10,b=20;
    a=&b;
    printf("%d\n",(*a+b));
    printf("%d\n",free(*a));
}


//Pointers to Functions
#include<stdio.h>
void ops(int a,int b,int *sum,int *prod,int *avg);
void main(){
    int a=10,b=10,sum,prod,avg;
    ops(a,b,&sum,&prod,&avg);
    printf("\n%d %d %d",sum,prod,avg);//20 100 10
```

```
}
void ops(int a,int b,int *s,int *p,int *av){
    *s=a+b;
    *p=a*b;
    *av=*s/2.0;
    printf("%d %d %d",*s,*p,*av);//20 100 10
}

//Pointer to array
#include<stdio.h>
void main()
{
    int a[]={10,20,30,40,50};
    int *ptr,i;
    ptr=a; // ptr=&a // ptr=&a[0];
    for(i=0;i<5;i++)
    {
        //printf("%d \n",&a[i]);
        printf("%d \n",ptr+i);//prints address location//ptr contains first address
value.As array is contiguous memory values soo as i has 4bytes size.ptr will
increase by 4 so that it can move to next as array is int.
        //printf("%d \n",a[i]);
        printf("%d \n\n",*(ptr+i));//10 20 30 40 50
    }
}

//Pointer Arithmetics
#include<stdio.h>
void main(){
    int n=8;
    char c='d';
    float f=9.8;
    int *nptr=&n;
    char *cptr=&c;
    float *fptr=&f;
    printf("Character pointer Before increment: %u After increment:
%u\n",cptr++,cptr);//Increment by 1(size of char)
    printf("Integer pointer Before increment: %u After increment:
%u\n",nptr++,nptr);//Increment by 4(size of int)
    printf("Float pointer Before increment: %u After increment: %u\n",fptr++,fptr);
//increment by 4(size of float)
}
/*Character pointer Before increment: 6422287 After increment: 6422288
```

```
Integer pointer Before increment: 6422288 After increment: 6422292
Float pointer Before increment: 6422280 After increment: 6422284*/

//Pointer Arithmetics
#include<stdio.h>
void main(){
   int n=8,m=9;
   int *ptr1=&n,*ptr2=&m;
   printf("%u and %u Difference=%u\n",ptr1,ptr2,ptr1-ptr2);//6422292 and 6422288
Difference=1
   printf("Comparison=%u\n",ptr1==ptr2);//Comparison=0
   printf("Comparison=%u",ptr1!=ptr2);//Comparison=1
}

//Dynamically inserting elements into array indirectly using pointers
#include<stdio.h>
void main(){
   int size;
   scanf("%d",&size);//3
   int arr[size];
   int *ptr=arr;
   for(int i=0;i<size;i++)
      scanf("%d",&(*ptr++));//2 3 4 or (ptr+i) or &arr[i]
   for(int i=0;i<size;i++)
      printf("%d ",arr[i]);//2 3 4 or *(ptr+i)
}
```

Day-4 Structures, Unions and DMA

```
//Structure example
#include <stdio.h>
#include<stdlib.h>
//Struct Creation
struct student
{
int rollno;
char name[30];//or *name for dynamic allocation //Need malloc function.Refer line
34.
float marks;
double fee;
};
```

```c
int main() {
    struct student stu1;
    struct student stu3;
    //Type-1 Initialization : individual
    stu1.rollno=12;
    stu1.marks=100.0;
    stu1.name[30]="Pranai D";//or strcpy(stu1.name,"Pranai D
sssdjdkkdsjishfshfohdhsosdjfs");//More than the size=>No problem it will print all
the name. //individual initialization means 'name[30]' should be written not only
'name'.
    //But in case of strcpy noneed to mention it.

    //Type-2 Initialization : As a group
    struct student stu2={10,"Pranai",99.12,123456};

    //Printing Values
    printf("Student 1 name is..%s\n",stu1.name);
    printf("Student 1 roll No:%d\n",stu1.rollno);
    printf("Student 1 marks:%.2f\n",stu1.marks);
    printf("Size of Roll No:%u\n",sizeof(stu1.rollno));
    printf("Name of student 2:%s\n",stu2.name);
    printf("Student 2 marks:%.2f\n",stu2.marks);

    //Type-3 Dynamic Initialization
    //Since string NAME is a pointer.Need to use Dynamic Memory Allocation in
case of Pointers
    stu3.name = (char*)malloc(100 * sizeof(char)); //*name for this type of
initialization.So it is showing error
    printf("Enter Roll No and name of student-3:");
    scanf("%d",&stu3.rollno);
    scanf("%[^\n]s",stu3.name);
    printf("\nName of student 3:%s\n",stu3.name);
    printf("Roll No of student-3:%d",stu3.rollno);

    //To avoid Memory Leaks and free the assigned remaining memory
    free(stu3.name);
    return 0;
}
/*
Student 1 name is..Pranai D
Student 1 roll No:12
Student 1 marks:100.00
Size of Roll No:4
```

```
Name of student 2:Pranai
Student 2 marks:99.12
Enter Roll No and name of student-3:25 Ratnam

Name of student 3: Ratnam
Roll No of student-3:25
*/

//Arroy of Structures
#include<stdio.h>
void main()
{
struct Student
{
char name[10];
int rollno;
float marks;
};
int n,i;
printf("How many students details you want: ");
scanf("%d",&n);
struct Student s[n]; // Arry of structures
for(i=0;i<n;i++)
{
printf("Enter student-%d details: ",i+1);
scanf("%s%d%f",s[i].name,&s[i].rollno,&s[i].marks);
}
printf("All students details are: \n");
for(i=0;i<n;i++)
{
printf("Name:%s \n",s[i].name);
printf("Roll Number:%d \n",s[i].rollno);
printf("Marks:%.2f \n",s[i].marks);
}
}
/*
How many students details you want: 2
Enter student-1 details: Pranai 12 365
Enter student-2 details: Hero 1 259
All students details are:
Name:Pranai
Roll Number:12
Marks:365.00
```

```
Name:Hero
Roll Number:1
Marks:259.00
*/
```

```
//Structures

//Dynamic Initialization
#include<stdio.h>
struct gita
{
 int sid;
 char name[10];
};
struct gita s1;//We can create a structure variable before main and after Structure creation
void main()
{
 printf("enter sid-1 : ");
 scanf("%d",&s1.sid);
 printf("enter sname-1 : ");
 scanf("%s",s1.name); // or strcpy(s1.name,"Pranai");
 printf("details:\n");
 printf("sid   : %d\n",s1.sid);
 printf("sname : %s\n",s1.name);
}
/*
enter sid-1 : 1
enter sname-1 : tarun
details:
sid : 1
sname : 6422292
*/

//Nested Structures
#include<stdio.h>
struct college
{
 int cid;
 char cname[20];
 struct dept
 {
   int did;
```

```
   char hname[20];
   float rating;
 }sd;
};
void main()
{
 //Initialization of NESTED Structures
 struct college sc = {101,"GIST - gangavaram",{301,"IT",3.6}};
 printf("\n");
 printf("college name       : %s\n",sc.cname);
 printf("college ID         : %d\n",sc.cid);
 printf("department ID      : %d\n",sc.sd.did);
 printf("HOD name           : %s\n",sc.sd.hname);
 printf("rating             : %f\n",sc.sd.rating);
}
/*
college name       : GIST - gangavaram
college ID         : 101
department ID      : 301
HOD name           : IT
rating             : 3.600000
*/

//Pointers to Structures
#include<stdio.h>
struct clg1
{
 int id;
 float marks;
};
struct clg1 *p,s;
void main()
{
 p=&s;
 printf("enter id :");
 scanf("%d",&p->id); // or  scanf("%d",&(*p).id);
 printf("ID: %d",p->id);   // or printf("ID: %d",(*p).id);
}

// Passing individual members of of a structure as parameters.
#include<stdio.h>
typedef struct
{
```

```
  char name[20];
  int age;
  float salary;
}EMPLOYEE;
void display(char n[],int a,float s);
void main()
{
  EMPLOYEE e;
  printf("Enter employee details: ");
  scanf("%s%d%f",e.name,&e.age,&e.salary);
  display(e.name,e.age,e.salary);
}
void display(char n[],int a,float s)
{
  printf("Name:%s \n",n);
  printf("Age:%d \n",a);
  printf("Salary:%.2f \n",s);
}
```

//Structures to functions

```
#include<stdio.h>
struct student{
  int roll;
  float cgpa;
  char name[30];
};
void details(struct student ds1){
  printf("RollNo-%d\n",ds1.roll);
  printf("CGPA-%.2f\n",ds1.cgpa);
  printf("Name-%s\n",ds1.name);
}
void main(){
  struct student s1={40,8.56,"Pranai D"};
  details(s1);
}
```

//UNIONS

```
#include<stdio.h>
union clg2
{
  //Both data types should be same//In case of different we get error
  float id;
  float marks;
```

```
};
union clg2 *p,s;
void main()
{
 p=&s;
 scanf("%f",&p->id);
 scanf("%f",&p->marks);
 printf("id :%f\n",p->id);
 printf("id :%f\n",p->marks);
}
```

```
//Dynamic Memory Allocation Example
#include <stdio.h>
#include <stdlib.h>

void main()
{
   // This pointer will hold the base address of the block created
   int* ptr;
   int n, i;

   // Get the number of elements for the array
   printf("Enter number of elements:");
   scanf("%d",&n);
   printf("Entered number of elements: %d\n", n);

   // Dynamically allocate memory using malloc()
   ptr = (int*)malloc(n*sizeof(int));
   for (i = 0; i < n; ++i) {
         scanf("%d",&ptr[i]);
      }
   // Print the elements of the array
   printf("The elements of the array are:\n");
   for (i = 0; i < n; ++i) {
      printf("%d\n", ptr[i]);
   }

   printf("Reallocating\n");
   printf("Enter resize of structure:");
   scanf("%d",&n);
   ptr = realloc(ptr,n);
   for (i = 0; i < n; ++i) {
         scanf("%d",&ptr[i]);
      }
```

```
    // Print the elements of the array
    printf("The elements of the array are:\n");
    for (i = 0; i < n; ++i) {
        printf("%d\n", ptr[i]);
    }
    free(ptr);
}

//malloc and calloc usage
#include <stdio.h>
#include<stdlib.h>
int main() {
    int n;
    int *ptr;
    scanf("%d",&n);
    ptr=(int *)malloc(n*sizeof(int)); //or (int *)calloc(n,sizeof(int));
    for(int i=0;i<n;i++){
        scanf("%d",(ptr+i)); //as it is a pointer as it contains address value.There is no
need to specify &(ambercent)
        printf("%u\n",*(ptr+i));//As it is a pointer we need to de reference it using
*(asterisk)
    }
    return 0;
}

//realloc usage
#include <stdio.h>
#include<stdlib.h>
int main() {
    int n;
    int *ptr;
    scanf("%d",&n);
    ptr=(int *)calloc(n,sizeof(int));
    for(int i=0;i<n;i++){
        scanf("%d",(ptr+i));
        printf("%u\n",*(ptr+i));
    }
    int m;
    printf("Enter resize value:");
    scanf("%d",&m);
    ptr=realloc(ptr,m);
    for(int i=0;i<m;i++){
        scanf("%d",(ptr+i));
```

```
        printf("%u\n",*(ptr+i));
    }
    return 0;
}

//Store 5 odd numbers using calloc and resize it to store 6 even numbers
//Realloc usage
#include <stdio.h>
#include<stdlib.h>
int main() {
    int i;
    int *ptr;
    ptr=(int *)calloc(5,sizeof(int));
    ptr[0]=1;
    ptr[1]=3;
    ptr[2]=5;
    ptr[3]=7;
    ptr[4]=9;
    for(int i=0;i<5;i++){
        printf("%u\n",*(ptr+i));
    }
    ptr=realloc(ptr,6);
    ptr[0]=2;
    ptr[1]=4;
    ptr[2]=6;
    ptr[3]=8;
    ptr[4]=10;
    ptr[5]=12;
    for(int i=0;i<6;i++){
        printf("%u\n",*(ptr+i));
    }
    free(ptr);
    return 0;
}
```

Day-5 Single Linked List

```
//Single Linked List Program

//Initially we create nn,st,cu as single first node.Refer cw for clarity.
//st means starting node which act as head that indicates first element in linked
list.
```

```c
//cu means current node that act as last element present in the linked list.
//te is temporary address/node that is used for traversal that starts from st to cu.

#include<stdio.h>
#include<stdlib.h>
//Template creation for linked list using structure
struct node{
   int data;
   struct node *next;
};
void create();
void display();
void insertionatbegin();
void insertionatend();
void insertatanyposition();
void deleteatbegin();
void deleteatend();
void deleteatanyposition();
struct node *nn,*te,*cu,*st=NULL;
void main(){
   int choice,option;
   do{
      printf("\nChoose options:\n");
      printf("1.Create.\n");
      printf("2.Display.\n");
      printf("3.Insertion at begin.\n");
      printf("4.Insertion at end.\n");
      printf("5.Insertion at any position.\n");
      printf("6.Deletion at Beginning.\n");
      printf("7.Deletion at End.\n");
      printf("8.Deletion at any position.\n");
      printf("9..Exit.\n");
      printf("Your option:");
      scanf("%d",&choice);
      switch(choice){
         case 1:
            create();
            break;
         case 2:
            display();
            break;
         case 3:
            insertionatbegin();
            break;
```

```
            case 4:
                insertionatend();
                break;
            case 5:
                insertatanyposition();
                break;
            case 6:
                deleteatbegin();
                break;
            case 7:
                deleteatend();
                break;
            case 8:
                deleteatanyposition();
                break;
            case 9:
                printf("\nCode Exited.\n");
                exit(0);
        }
        printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
        scanf("%d",&option);
    }while(option!=0);
}
void create(){
    nn=(struct node*)malloc(sizeof(struct node *));
    printf("Enter node value:");
    scanf("%d",&(nn->data));
    nn->next=NULL;//As new node is created its address will be set as NULL.It is also
a cu but it will become cu at line 51(updating previous cu with nn that is created
now)if elements are already present in list otherwise line 47.
    if(st==NULL){
        st=nn;
        cu=nn;
    }
    else{
        cu->next=nn;//Previous cu next contains present created new node address so
that previous will be linked to new created nn
        cu=nn;//so as previous node is linked with nn in above step.Now we need to
change nn as cu.so that it can be used for next linkage.
        //As this is self referential structure cc and nn act as address so we store next
value of cu as address of nn directly by nn but not nn->next.
    }
    printf("Node created.\n");
}
```

```
void display(){
   if(st==NULL)//st is null means list is empty as it is set null during initialization of
program.
      printf("Linked List is Empty.\n");
   else{
      int i=1;
      te=st;//te should start from first node.so we assign st to te.
      printf("Elements are...\n");
      while(te!=NULL){
         printf("%d-->",te->data);
         te=te->next;//Moving from one node to next node till it reaches cu i.e.,next
becomes null as cu will be having null.
      }
   }
}
void insertionatbegin(){
   printf("Before insertion at begin:\n");
   display();
   nn=(struct node*)malloc(sizeof(struct node *));
   printf("\nEnter node value:");
   scanf("%d",&(nn->data));
   if(st==NULL){
      nn->next=NULL;
      st=nn;
      cu=nn;
      display();
   }
   else{
      nn->next=st;
      st=nn;
      printf("After insertion at begin:\n");
      display();
   }
}
void insertionatend(){
   printf("Before insertion at end:\n");
   display();
   nn=(struct node*)malloc(sizeof(struct node *));
   printf("Enter node value:");
   scanf("%d",&(nn->data));
   if(st==NULL){
      nn->next=NULL;
      st=nn;
      cu=nn;
```

```
        display();
    }
    else{
        cu->next=nn;
        nn->next=NULL;
        printf("After insertion at end:\n");
        display();
    }
}
void insertatanyposition(){
    printf("Before insertion at any position:\n");
    display();
    nn=(struct node*)malloc(sizeof(struct node *));
    printf("\nEnter node value:");
    scanf("%d",&(nn->data));
    if(st==NULL){
        nn->next=NULL;
        st=nn;
        cu=nn;
        display();
    }
    else{
        int pos;
        printf("Enter position where to be inserted:");
        scanf("%d",&pos);
        int count=1;
        te=st;
        while(te!=NULL){
            count++;
            if(count==pos){
                nn->next=te->next;
                te->next=nn;
                printf("After insertion at any position:\n");
                display();
                break;
            }
            else if(count<pos){
                nn->next=NULL;
                printf("List reached end.Cannot insert at your required position.\n");
                break;
            }
            else
                continue;
            te=te->next;
```

```
        }
    }
}
void deleteatbegin(){
    if(st==NULL)
        printf("List is empty cannot perform delete operation.\n");
    else{
        printf("Before Deletion.\n");
        display();
        te=st->next;
        st=te;
        printf("After Deletion.\n");
        display();
    }
}
void deleteatend(){
    if(st==NULL)
        printf("List is empty cannot perform delete operation.\n");
    else{
        printf("Before Deletion.\n");
        display();
        te=st;
        while(te->next->next!=NULL){
            te = te->next;
        }
        te->next = NULL;
        printf("After Deletion.\n");
        display();
    }
}
void deleteatanyposition(){
    int pos;
    printf("Enter position of element to be deleted:");
    scanf("%d",&pos);
    if(st==NULL)
        printf("List is empty cannot perform delete operation.\n");
    else{
        struct node *te1,*te2;
        te1=st;
        te2=st->next;
        printf("Before Deletion.\n");
        display();
        for(int i=2;i<=pos;i++){
            if(i==pos){
```

```
            te1->next=te2->next;
            te2->next=NULL;
            printf("\nAfter Deletion.\n");
            display();
            break;
        }
        te1=te2;
        te2=te2->next;
    }
  }
}
```

Day-6 Stacks & Circular, Double Linked List

```
//circular linked list
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
void create();
void display();
struct node *nn,*st=NULL,*cu,*te;
void main(){
   int choice,option;
   do{
      printf("\nChoose options:\n");
      printf("1.Create.\n");
      printf("2.Display.\n");
      printf("3.Exit.\n");
      printf("Your option:");
      scanf("%d",&choice);
      switch(choice){
         case 1:
            create();
            break;
         case 2:
            display();
            break;
         case 3:
            printf("\nCode Exited.\n");
```

```
            exit(0);
        }
        printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
        scanf("%d",&option);
    }while(option!=0);
}
void create(){
    nn=(struct node*)malloc(sizeof(struct node *));
    printf("Enter node value:");
    scanf("%d",&(nn->data));
    nn->next=NULL;
    if(st==NULL){
        st=nn;
        cu=nn;
    }
    else{
        cu->next=nn;
        cu=nn;
        nn->next=st;
        }
    printf("Node created.\n");
}
void display(){
    if(st==NULL)
        printf("Linked List is Empty.\n");
    else{
        int i=1;
        te=st;
        printf("Elements are...\n");
        do{
            printf("%d-->",te->data);
            te=te->next;
        }while(te->next!=st->next);
    }
}
```

```
//Double linked list
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next,*prev;
};
```

```
void create();
void display();
struct node *cu,*nn,*te,*st=NULL;
void main(){
    int choice,option;
    do{
        printf("\nChoose options:\n");
        printf("1.Create.\n");
        printf("2.Display.\n");
        printf("3.Exit.\n");
        printf("Your option:");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                create();
                break;
            case 2:
                display();
                break;
            case 3:
                printf("\nCode Exited.\n");
                exit(0);
        }
        printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
        scanf("%d",&option);
    }while(option!=0);
}
void create(){
    nn=(struct node*)malloc(sizeof(struct node *));
    printf("Enter node value:");
    scanf("%d",&(nn->data));
    nn->next=NULL;
    if(st==NULL){
        nn->prev=NULL;
        st=nn;
        cu=nn;
    }
    else{
        nn->prev=cu;
        cu->next=nn;
        cu=nn;
    }
    printf("Node created.\n");
}
```

```c
void display(){
    //Forward
    int bf;
    printf("Forward(1) or Backward(0):");
    scanf("%d",&bf);
    if(bf==1){
        if(st==NULL)
            printf("Linked List is Empty.\n");
        else{
            struct node *fte;
            fte=st;
            printf("Elements are...\n");
            while(fte!=NULL){
                printf("%d-->",fte->data);
                fte=fte->next;
            }
        }
    }
    else{
        //Backward
        if(st==NULL)
            printf("Linked list is Empty.\n");
        else{
            struct node *bte;
            bte=cu;
            printf("Elements are...\n");
            while(bte->prev!=NULL){
                printf("%d-->",bte->data);
                bte=bte->prev;
            }
        }
    }
}
```

```c
//Stacks using linkedlist
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *nn,*st=NULL,*cu,*top,*stop,*atop;
void push();
```

```
void display();
void pop();
void sum();
int acount();
void average();
void main(){
   int choice,option;
   do{
      printf("\nChoose options:\n");
      printf("1.Push.\n");
      printf("2.Display.\n");
      printf("3.Pop\n");
      printf("4.Sum.\n");
      printf("5.Average.\n");
      printf("6.Exit.\n");
      printf("Your option:");
      scanf("%d",&choice);
      switch(choice){
         case 1:
            push();
            break;
         case 2:
            display();
            break;
         case 3:
            pop();
            break;
         case 4:
            sum();
            break;
         case 5:
            average();
            break;
         case 6:
            printf("\nCode Exited.\n");
            exit(0);
      }
      printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
      scanf("%d",&option);
   }while(option!=0);
}
void push(){
   nn=(struct node *)malloc(sizeof(struct node *));
   printf("Enter data:");
```

```
    scanf("%d",&(nn->data));
    nn->next=st;
    st=nn;
    printf("Element pushed successfully.\n");

}
void display(){
    if(st==NULL)
        printf("Stack is empty.");
    else{
        top=st;
        while(top!=NULL){
            printf("%d\n",top->data);
            top=top->next;
        }
    }
}
void pop(){
    if(st==NULL)
        printf("Stack is empty.");
    else{
        st=st->next;
        printf("Element popped successfully.\n");
    }
}
void sum(){
    if(st==NULL)
        printf("Stack is empty.Sum cannot be performed.\n");
    else{
        stop=st;
        int sum=0;
        while(stop!=NULL){
            sum=sum+(stop->data);
            stop=stop->next;
        }
        printf("Sum is %d.",sum);
    }
}
int acount(){
    int count=0;
    if(st==NULL)
        printf("Stack is empty.Average cannot be performed.\n");
    else{
        top=st;
```

```
        while(top!=NULL){
            count++;
            top=top->next;
        }
    }
    return count;
}
void average(){
    float sum=0;
    if(st==NULL)
        printf("Stack is empty.Average cannot be performed.\n");
    else{
        atop=st;
        while(atop!=NULL){
            sum+=atop->data;
            atop=atop->next;
        }
    }
    printf("%.2f",sum/acount());
}
```

Day-6 HW Calculator using stack

```
// Calculator using Stack
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *nn,*st=NULL,*cu,*top,*stop,*atop,*sstop,*subsi;
void push();
void display();
void pop();
void add();
void sub();
int acount();
void average();
void mul();
void cdiv();
void main(){
    int choice,option;
```

```
printf("\n=============Calculator using stack================");
do{
    printf("\nChoose options:\n");
    printf("1.Push.\n");
    printf("2.Pop.\n");
    printf("3.Display\n");
    printf("4.Addition.\n");
    printf("5.Subtraction.\n");
    printf("6.Multiplication.\n");
    printf("7.Division.\n");
    printf("8.Average.\n");
    printf("9.Exit.\n");
    printf("Your option:");
    scanf("%d",&choice);
    switch(choice){
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            display();
            break;
        case 4:
            add();
            break;
        case 5:
            sub();
            break;
        case 6:
            mul();
            break;
        case 7:
            cdiv();
            break;
        case 8:
            average();
            break;
        case 9:
            printf("\nCode Exited.\n");
            exit(0);
    }
    printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
```

```
        scanf("%d",&option);
    }while(option!=0);
}
void push(){
    nn=(struct node *)malloc(sizeof(struct node *));
    printf("Enter data:");
    scanf("%d",&(nn->data));
    nn->next=st;
    st=nn;
    printf("Element pushed successfully.\n");

}
void display(){
    if(st==NULL)
        printf("Stack is empty.");
    else{
        printf("Stack Elements are...\n");
        top=st;
        while(top!=NULL){
            printf("%d\n",top->data);
            top=top->next;
        }
    }
}
void pop(){
    if(st==NULL)
        printf("Stack is empty.");
    else{
        st=st->next;
        printf("Element popped successfully.\n");
    }
}
int acount(){
    int count=0;
        top=st;
        while(top!=NULL){
            count++;
            top=top->next;
        }
    return count;
}
void average(){
    float sum=0;
    if(st==NULL)
```

```
        printf("Stack is empty.Average cannot be performed.\n");
    else{
        atop=st;
        while(atop!=NULL){
            sum+=atop->data;
            atop=atop->next;
        }
        printf("%.2f",sum/acount());
    }
}
void add(){
    if(st==NULL)
        printf("Stack is empty.Sum cannot be performed.\n");
    else{
        stop=st;
        int sum=0;
        while(stop!=NULL){
            sum=sum+(stop->data);
            stop=stop->next;
        }
        printf("Sum is %d.",sum);
    }
}
void sub(){
    int subs=0;
    if(st==NULL){
        printf("Stack is empty.Subtraction cannot be performed.\n");
    }
    else{
        sstop=st;
        subsi=st->next;
        subs=(sstop->data)-(subsi->data);
        while(subsi->next!=NULL){
            subsi=subsi->next;
            subs=subs-(subsi->data);
        }
    printf("Subtraction is %d.",subs);
    }
}
void cdiv(){
    float divs=0;
    if(st==NULL){
        printf("Stack is empty.Division cannot be performed.\n");
    }
```

```
    else{

        sstop=st;
        subsi=st->next;
        divs=(sstop->data)/(subsi->data);
        while(subsi->next!=NULL){
            subsi=subsi->next;
            divs=divs/(subsi->data);
        }
    printf("Division is %.2f.",divs);
    }
}
void mul(){
    int mul=0;
    if(st==NULL){
        printf("Stack is empty.Multiplication cannot be performed.\n");
    }
    else{

        sstop=st;
        subsi=st->next;
        mul=(sstop->data)*(subsi->data);
        while(subsi->next!=NULL){
            subsi=subsi->next;
            mul=mul*(subsi->data);
        }
    printf("Multiplication is %d.",mul);
    }
}
```

Day-7 Queue

```
//Queue using structures
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *fe = NULL, *re = NULL, *te, *nn, *cu;
void enqueue();
```

```
void dequeue();
void display();

void main()
{
   int choice, option;
   do
   {
      printf("\nChoose options:\n");
      printf("1.Enqueue.\n");
      printf("2.Dequeue.\n");
      printf("3.Display.\n");
      printf("4.Exit.\n");
      printf("Your option:");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
         enqueue();
         break;
      case 2:
         dequeue();
         break;
      case 3:
         display();
         break;
      case 4:
         printf("\nCode Exited.\n");
         exit(0);
      }
      printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
      scanf("%d", &option);
   } while (option != 0);
}
void enqueue()
{
   nn = (struct node *)malloc(sizeof(struct node *));
   printf("Enter data value:");
   scanf("%d", &(nn->data));
   nn->next = NULL;
   if (re == NULL && fe == NULL)
   {
      fe = re = cu = nn;
   }
```

```
    else
    {
        nn->next = cu;
        fe = nn;
        cu = nn;
    }
    printf("Data enqueued successfully");
}
void dequeue()
{
    if (re == NULL && fe == NULL)
    {
        printf("Queue is empty.Can't perform deletion");
    }
    else
    {
        te = fe;
        if (te->next == NULL)
        {
            re = fe = NULL;
        }
        else
        {
            while (te->next->next != NULL)
            {
                te = te->next;
            }
            te->next = NULL;
            re = te->next;
            printf("Data dequeued successfull.");
        }
    }
}
void display()
{
    te = fe;
    if (re == NULL && fe == NULL)
        printf("Queue is empty.\n");
    else
    {
        printf("Queue is..\n");
        while (te != NULL)
        {
            printf("<--%d", te->data);
```

```
            te = te->next;
        }
    }
}
```

Day-9 Binary Search Tree

```
//Binary Search Tree
#include<stdio.h>
#include<stdlib.h>
struct node {
    int data;
    struct node *left;
    struct node *right;
};

void create();
void display();

struct node *nn,*rn=NULL,*cu;

void main(){
    int choice, option;
    do
    {
        printf("\nChoose options:\n");
        printf("1.Create/Insert.\n");
        printf("2.Display.\n");
        printf("3.Exit.\n");
        printf("Your option:");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            create();
            break;
        case 2:
            display();
            break;
        case 3:
            printf("\nCode Exited.\n");
            exit(0);
        }
```

```
        printf("\nDO YOU WANT TO CONTINUE(1) OR NOT(0):");
        scanf("%d", &option);
    } while (option != 0);
}

void create(){
    nn=(struct node*)malloc(sizeof(struct node *));
    printf("Enter data value:");
    scanf("%d",&(nn->data));
    nn->left=NULL;
    nn->right=NULL;
    if(rn==NULL){
        rn=cu=nn;
    }
    else{
        cu=rn;
        while(cu->left!=NULL || cu->right!=NULL){
            printf("hshhs");
            if(nn->data < cu->data){
                printf("hsdsud");
                cu=cu->left;
            }
            else{
                printf("shdfosijf");
                cu=cu->right;
            }
        }
        if(nn->data < cu->data){
                cu->left=nn;
        }
        else{
            cu->right=nn;
        }
    }
}

void display(){
    cu=rn;
    printf("%d\n",cu->data);
    cu=cu->left;
    printf("%d\n",cu->data);
    cu=cu->left;
    printf("%d\n",cu->data);
    cu=cu->left;
```

```
    printf("%d\n",cu->data);
    cu=cu->left;
    printf("%d\n",cu->data);
}
```

Day-12 Searching and Sorting

```
// Jump Search
#include <stdio.h>
#include<math.h>
int main() {
    int n;
    printf("Enter size of array:");
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
    int i=0;
    int j=0;
    int k;
    int jmp=sqrt(n);
    printf("%d\n",jmp);
    printf("Enter element to be found:");
    scanf("%d",&k);
    while(j<=n){
        if(k==arr[j]){
            printf("Element found at index %d",j);
            break;
        }
        else if(k>arr[j]){
            j+=jmp;
        }
        else if(k<arr[j]){
            j--;
        }
    }
    if(j>n)
        printf("Element not found");
    return 0;
}

// Bubble Sort
```

```c
#include <stdio.h>
int main() {
    int n;
    printf("Enter size of array:");
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
    int swap=0,temp;
    for(int i=0;i<n;i++){
        for(int j=0;j<n-1-i;j++){
            if(arr[j]>arr[j+1]){
                swap+=1;
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
    printf("No of swaps done:%d\n",swap);
    printf("Sorted elements are...");
    for(int i=0;i<n;i++)
        printf("%d ",arr[i]);
    return 0;
}
```

//Selection sort

```c
#include <stdio.h>
int main() {
    int n;
    printf("Enter size of array:");
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
    int temp;
    for(int i=0;i<n;i++){
        int small=arr[i];
        int index=0,c=0;
        for(int j=i+1;j<n;j++){
            if(arr[j]<small){
                small=arr[j];
                index=j;
                c+=1;
```

```
        }
      }
      if(c==0){
         continue;
      }
      else{
         temp=arr[i];
         arr[i]=small;
         arr[index]=temp;
      }
   }
   printf("Sorted elements are...");
   for(int i=0;i<n;i++)
      printf("%d ",arr[i]);
   return 0;
}
```

Extra Practice Programs

```
/*Problem statement:
Sushi has 3 numbers A,B and C.
Sushi wonders if it is possible to choose exactly two numbers out of the three
numbers such that theit sum is odd.
Input Format:
• The first line of input will contain a single integer T
• T, denoting the number of test cases.
• Each test case consists of three integers A,B,C
Output Format:
YES if possible
NO if not possible

written in python
*/
s=int(input())
n=y=0
for i in range(s):
   lis-list(map(int,input().split('')))
   c=[]
   for j in range(len(lis)):
      if (sum(lis)-lis[j])%2!=0:
         c.append(1)
      else:
```

```
        c.append(0)
    if 1 in c:
        print("YES")
    else:
        print("NO")
/*
```
Problem statement:

In Island, denominations less than rupees 10 have stopped and now rupees 10 is the smallest denomination.
Suppose sushi goes to buy some item with cost not a multiple of, then, he will be charged the cost that is the nearest multiple of 10. If the cost is equally distant from two nearest multiples of 10, then the cost is rounded up For example, 35,38,40,44, 35,38,40,44 are all rounded to 40.
Suppose sushi purchased an item of price x(x<100).How much will be given as change if sushi gave 100 to the seller

```
*/
t=int(input())
lis=[]
for i in range(t):
    lis.append(int(input()))
for i in lis:
    if i%10>=5:
        print(100-(i+(10-i%10)))
    else:
        print(100-(i-(i%10)))
```

//print the desired nth prime number

```
//ex: ip=5
//    op=11
#include<stdio.h>
void main(){
    int num=0;
    int a;
    scanf("%d",&a);
    for(int i=2;i<1000;i++){
        int count=0;
        for(int j=1;j<=i;j++){
            if(i%j==0)
                count++;
        }
        if(count==2){
            num++;}
        if(num==a){
```

```
            printf("%d",i);
            break;}
    }
}

//Store a value at a variable indirectly using pointer
#include<stdio.h>
void main(){
    int *ptr,x;
    ptr=&x;
    printf("%u\n%u\n",&x,ptr);
    *ptr=10;//indirect assignment of 10 to variable x.
    printf("x=%d",x);
}


//Malloc and Reaalloc Example
#include <stdio.h>
#include <stdlib.h>
int main()
{
    // This pointer will hold the
    // base address of the block created
    int* ptr;
    int m,n, i;
    // Get the number of elements for the array
    scanf("%d",&m);
    // Dynamically allocate memory using calloc()
    ptr = (int*)calloc(m, sizeof(int));
    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        // Get the elements of the array
        for (i = 0; i < m; ++i) {
            scanf("%d",ptr+i);
        }
        // Print the elements of the array
        printf("Initial array: ");
        for (i = 0; i < m; ++i) {
```

```
            printf("%d ", ptr[i]);
        }
        printf("\n");
        // Get the new size for the array
        scanf("%d",&n);
        // Dynamically re-allocate memory using realloc()
        ptr = (int*)realloc(ptr, (n+m) * sizeof(int));
        // Get the new elements of the array
        for (i = m; i < (n+m); ++i) {
            scanf("%d",(ptr+i));
        }
        printf("Total elements processed: %d\n",m);
        // Print the elements of the array
        printf("Reallocated Array: ");
        for (i = 0; i < (n+m); ++i) {
            printf("%d ", ptr[i]);
        }
        printf("\nTotal elements processed: %d\n",m+n);
        free(ptr);
    }
    return 0;
}
```

```
#Find whether given number is present in fibonacci series or not
def fib(num):
    if num==1 or num==0:
        return num
    else:
        return fib(num-1)+fib(num-2)
num=int(input())
lis=[]
for i in range(num+1):
    if fib(i)==num:
        lis.append(1)
    else:
        lis.append(0)
if 1 in lis:
    print("Yes")
else:
    print("No")
```