Tamal Datta

1) WAP to set all the bits which are having position multiple of 7 from 9th position for given
integer using bitwise operator.
EX: i/p: 156
00000000000000000000000010011100
o/p: 00010000001000000100000010011100

Solution:

// **C** //

```
#include<stdio.h>
void main()
{
   int n;
   puts("enter the number");
   scanf("%d",&n);
   printf("entered number is %d\n",n);
   puts("before modifing");
   for(int i=31;i>=0;i--)
      printf("%d",n>>i&1);

   for(int i=31;i>=0;i--)
      if(!(i%7) && i>9)
         n=n|1<<i;
   puts("after modifying");
   for(int i=31;i>=0;i--)
      printf("%d",n>>i&1);
 }
```

// **Python** //

```
n=int(input("enter number "))
s='{:032b}'.format(n)
print("before modifying :",s)
s=list(s[::-1])
i=0
while i<len(s):
   if (i%7==0) and i>9:
      s[i]="1"
   i=i+1
s="".join(s[::-1])
print("after modifying :",s)
```

2) WAP to print following pattern.

```
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
```

NOTE: create generic code for pattern without using for loop.

Solution:

// **C** //

```
#include<stdio.h>
void main()
{
   int n;
   puts("n=??");
   scanf("%d",&n);
   int i=0,j;
   while(i<n)
      {
         j=0;
         while(j<=n-i-1)
            {printf("  "); j++; }
         j=1;
         while(j<=i+1)
            {printf("%d ",j); j++; }
         j=i;
         while(j>0)
            {printf("%d ",j); j--; }
         puts("");
      i++;
      }
}
```

// **Python** //

```
n=int(input("enter number "))
out=[]
for i in range(1,n+1):
   out=list(range(1,i+1))+list(range(i-1,0,-1))
   print(" "*(n-i),end="")
   for ch in out:
      print(ch,end="")
   print()
```

3) WAP to check whether a given number is a perfect cube or not.
EX: i/p: 125
The number is a perfect Cube of 5
Solution:

```
////////////////////////////////////////////////// C //////////////////////////////////////////////////
#include<stdio.h>
void main()
{
   int n;
   puts("enter the number");
   scanf("%d",&n);
   int k=1;
   while(k<n/2)
      {
         if(k*k*k==n)
            {
               printf("%d is perfect cube of %d\n",n,k);
               return;
            }
         k++;
      }
   printf("%d is not a perfect cube\n",n);
}
////////////////////////////////////////////////// Python //////////////////////////////////////////////////
n=int(input("enter number "))
for i in range(n//2):
   if i**3==n:
      print("{} is perfect cube of {}".format(n,i))
      break
else:
   print("{} is not a perfect cube".format(n))
```

4) WAP to find a pair with given sum in the given array.
EX: The given array : 6 8 4 -5 7 9
The given sum : 15
Pair of elements can make the given sum by the value of index 0 and 5
Solution:

```
////////////////////////////////////////////////// C //////////////////////////////////////////////////
#include<stdio.h>
void main()
{
   int n;
   puts("no. of elements of array?");
```

```
    scanf("%d",&n);
    puts("enter the array");
    int arr[n];
    for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
    puts("enter the sum value");
    int s,f=0;
    scanf("%d",&s);

    for(int i=0;i<n-1;i++)
        for(int j=i+1;j<n;j++)
            if(arr[i]+arr[j]==s)
                {
                    printf("index of pair elements are %d %d\n",i,j);
                    f=1;
                }
    if(f==0)
        puts("no pair element is present");
}
```

// **Python** //

```
n=int(input("enter number of elements?"))
l=[]
print("enter elements")
for i in range(n):
  l.append(int(input()))
s=int(input("enter sum value"))

for i in l:
    if s-i in l:
        k1=l.index(i)
        k2=l.index(s-i)
        print("pair elements are {} {}".format(k1,k2))
        break
else:
    print("pair elements not present")
```

5) WAP to find the missing number from a given array. There are no duplicates in list.
EX: The given array is : 1 3 4 2 5 6 9 8
The missing number is : 7

Solution:

// **C** //

```
#include<stdio.h>
void main()
{
   int n;
   puts("no. of elements of array?");
   scanf("%d",&n);
   puts("enter the array");
   int arr[n],sum=0;
   for(int i=0;i<n;i++)
      {
         scanf("%d",&arr[i]);
         sum+=arr[i];
      }

   for(int i=0;i<n-1;i++)
      for(int j=i+1;j<n;j++)
         if(arr[j]<arr[i])
            arr[i]=arr[j]-arr[i]+(arr[j]=arr[i]);

   int nsum=(arr[n-1]*(arr[n-1]+1)/2 )+1 - (arr[0]*(arr[0]+1)/2 );

   if(nsum-sum==0)
      puts("no number is missing");
   else
      printf("missing number is %d\n",nsum-sum);

}
```

// **Python** //

```
n=int(input("enter number of elements?"))
l=[]
print("enter elements")
for i in range(n):
  l.append(int(input()))
l=sorted(l)
for i in range(l[0],l[len(l)-1]):
   if i not in l:
      print("missing number is ",i)
      break
else:
   print("no number is missing")
```

6) WAP to insert two extra elements in given array, increment first element by one and insert
before it and decrement last element by one and insert after it using function.
EX: The given array : 10 20 30 40 50
o/p array: 11 10 20 30 40 50 49
NOTE: use this prototyp: int my_fun(int*);

Solution:

// **C** //

```
#include<stdio.h>
#include<stdlib.h>
int* my_fun(int *arr);
int l=0;

void main()
{
   int n;
   puts("enter the length of array");
   scanf("%d",&n);
   int arr[n];
   l=n;
   puts("enter elements of array");
   for(int i=0;i<n;i++)
      scanf("%d",&arr[i]);

   puts("old array is:");
   for(int i=0;i<n;i++)
      printf("%d  ",arr[i]);
   int* arr1=my_fun(arr);

   puts("new array is:");
   for(int i=0;i<n+2;i++)
      printf("%d  ",*(arr1+i));

}

int* my_fun(int *arr)
{
   int* a=malloc(4*(l+2)); // it must be dynamic else segmentation fault
   int i=0;
   for(i=0;i<=l;i++)
      {
         if(i==0)
            {
               a[i]=arr[0]+1;
               continue;
            }
```

```
        a[i]=arr[i-1];
        }
    a[i]=arr[i-2]-1;

    return a;
}
```

// **Python** //

```
def my_fun(l):
    l1=[]
    l1[1:-2]=l
    l1.insert(0,l[0]+1)
    l1.insert(-1,l[-1]-1)
    return l1

n=int(input("enter number of elements?"))
l=[]
print("enter elements")
for i in range(n):
   l.append(int(input()))
print("old array is:")
for i in l:
   print(i,end=" ")
print()
k=my_fun(l)
print("new array is:")
for i in k:
   print(i,end=" ")
```

7) WAP to find maximum occurring character in a string.

EX: i/p: ”vector india pvt ltd”

The maximum occurring character in a string: 't' appears number of times: 3

Solution:

// **C** //

```
#include<stdio.h>
#include<string.h>
void main()
{
    char s1[20],s[20];
    puts("enter the string");
    gets(s1);
    strcpy(s,s1);
    int i=0,v=0,count=1,j=0;
    char k;
    while(1)
        {
```

```
        if(s[0]==0x20)  // to avoid the situation if str starts with 0x20
           {
              strcpy(&s[0],&s[1]);
              continue;
           }
        for(i=1;s[i];i++)
           {
           L:   if(s[i]==0x20)    // for avoiding 0x20
                {
                strcpy(&s[i],&s[i+1]);
                goto L;
                }

              if(s[0]==s[i])
                {
                count++;
                strcpy(&s[i],&s[i+1]);
                i--;
                }
           }
        if(count>=v)
           {
              v=count;
              k=s[0];
           }
        if(s[0]==0)
           break;
        strcpy(&s[0],&s[1]);
        count=1;

     }
   printf("%c -> char has occured maximum, %d times\n",k,v);
}
```

// **Python** //

```
s=input("enter the string: ")
d={}
for i in s:
   if i!=" ":
      d[i]=d.get(i,0)+1
x=0
for k,v in d.items():
   if v>=x:
      x=v
      ch=k
print("{}->char occured maximum,{} times".format(ch,x))
```

8) WAP to Print Fibonacci Series using recursion.

Solution:

// **C** //

```
#include<stdio.h>
void my_fun(int* f,int l);

void main()
{
   int n;
   printf("enter length of series\n");
   scanf("%d",&n);
   int f[n];
   my_fun(f,sizeof(f)/sizeof(f[0]));

   printf("first %d fibonacci number are :\n",n);
   for(int i=0;i<n;i++)
      printf("%d  ",f[i]);

}

void my_fun(int* f,int l)
{

   static int i=2;
   f[0]=1;
   f[1]=1;
   if(i<l)
      {
      f[i]=f[i-1]+f[i-2];
      i++;
      my_fun(f,l);
      }
}
```

// **Python** //

```
n=int(input("enter no. of terms"))
l=[]
l.append(1)
l.append(1)
i=2
while i<n:
   l.append(l[i-1]+l[i-2])
   i=i+1

for i in l:
   print(i,end=" ")
```

9) WAP to check the given number is palindrome or not using recursion.
NOTE: use this prototyp: int my_fun(int*);

Solution:

// **C** //

```
#include<stdio.h>
int my_fun(int* n);
void main()
{
   int n;
   puts("enter the number");
   scanf("%d",&n);
   printf("%d is ",n);
   my_fun(&n)?puts("a palindrome number"):puts("not a palindrome number");
}

int my_fun(int* n)
{
   static int flag=0,sum=0,sum1=0;
   if(flag==0)
      {
      sum1=*n;
      flag=1;
      }
   if(*n!=0)
   {
      sum=10*sum+(*n)%10;
      *n=*n/10;
      my_fun(n);
    }
return sum==sum1?1:0;
}
```

// **Python** //

```
n=(input("enter the number "))
out="palindrome" if n==n[::-1] else "not palindrome"
print(n,"is",out)
```

10) WAP to delete given char from given string using recursion.
EX: i/p: ”vector india pvt ltd”
o/p: “vecor india pv ld”
NOTE: use this prototyp: char* my_fun(char*,char);

Solution:

// **C** //

```
#include<stdio.h>
#include<string.h>
char* my_fun(char* s,char c);
```

```
void main()
{
   char s[20],c;
   puts("enter the string");
   gets(s);
   puts("enter the character to delete");
   scanf("%c",&c);
   char *s1=my_fun(s,c);
   puts("after deletion the char new string is:");
   puts(s1);
}

char* my_fun(char* s,char c)
{
   if(*s)
      {
         if(*s==c)
            {
             strcpy(s,s+1);
             my_fun(s,c);
            }
         else
             my_fun(s+1,c);
      }
   return s;
}
```

// **Python** //

```
def my_fun(s,c):
   out=''
   for i in s:
      if i is not c:
         out+=i
   return out

s=input("enter the string :")
c=input("enter char to delete :")
print("old string is:\t",s)
s=my_fun(s,c)
print("new string is:\t",s)
```