PROBLEM 1(KAPREKAR NUMBER)

```
#include <stdio.h>
#include<math.h>
int main(int argc, char const *argv[])
{
	int n;
	int temp;
	int s;
	int flag;
	flag=0;
	scanf("%d",&n);
	s=n*n;
	temp=s;
	int count;
	count=0;
	while(s>0)
	{
		s=s/10;
		++count;
	}
	s=temp;
	for(int i=1;i<=count-1;++i)
	{
		int d;
		d=pow(10,i);
		if(d==n)
		{
			continue;
		}
		int sum;
		sum=s/d + s%d;
		if(sum==n)
		{
			flag=1;
			break;
		}
	}
	if(flag==1)
	{
		printf("It is kaprekar number");
	}
	else
	{
		printf("not a kaprekar number");

	}
	return 0;
```

}

OUTPUT:

45
It is kaprekar number

FUNCTIONS

IT IS A SET OF INSTRUCTIONS GOING TO WRITE INSIDE A BLOCK AT ONCE WHICH CAN BE USED MULTIPLE TIMES IT ENSURES CODE REUSABILITY

PROGRAMMING PARADIGMS

1. MONOLITHIC PROGRAMMING (WITHIN THE MAIN FUNCTION)
2. MODULAR PROGRAMMING (FUNCTIONAL OR STRUCTURAL)
3. OBJECT ORIENTED PROGRAMMING

FUNCTIONS TYPES:

1. NO ARGUMENT NO RETURN TYPE
2. NO ARGUMENT WITH RETURN TYPE
3. WITH ARGUMENT NO RETURN TYPE
4. WITH ARGUMENT WITH RETURN TYPE

DECLARATION OF FUNCTION:

MAIN ()-→USING OPEN AND CLOSE BRACKETS TO DECLARE FUNCTIONS

- **NO ARGUMENT NO RETURN TYPE:**

```
 #include <stdio.h>
void sum()
{
        int a;
        int b;
        printf("enter the number a");
        scanf("%d",&a);
        printf("enter the number b");
        scanf("%d",&b);
        printf("%d", a +b);
}
int main (int argc, char const *argv[])
{
        Sum ();
        return 0;
}
```

- **NO ARGUMENT WITH RETURN TYPE:**

```
#include <stdio.h>
int sum()
{
        int a;
        int b;
        printf("enter the number a");
        scanf("%d",&a);
        printf("enter the number b");
        scanf("%d",&b);
        return a+b;
}
int main(int argc, char const *argv[])
{
        int result = sum();
        printf("%d",result);
        return 0;
}
```

- **WITH ARGUMENT NO RETURN TYPE:**

```
#include <stdio.h>
void sum(int a, int b)
{
        printf("%d",a+b);
}
int main(int argc, char const *argv[])
{
        int a;
        int b;
        printf("enter the number a");
        scanf("%d",&a);
        printf("enter the number b");
        scanf("%d",&b);
        sum(a,b);

        return 0;
}
```

- **WITH ARGUMENT WITH RETURN TYPE:**

```
#include <stdio.h>
int sum(int m, int n)
{
        return(m+n);
}
int main(int argc, char const *argv[])
```

```
{
        int m;
        int n;
        printf("enter the number a:\n");
        scanf("%d",&m);
        printf("enter the number b:\n");
        scanf("%d",&n);
        int r = sum(m,n);
        printf("%d",r);
        return 0;
}
```

ADAM NUMBER USING FUNCTIONS:

```
#include<stdio.h>
int sqr(int n )
{
   return n*n;
}
int reverse(int n)
{
   int rem;
   int r= 0;
   while(n>0)
   {
      rem=n%10;
      r=r*10+rem;
      n=n/10;
   }
   return r;
}
int main(int argc, char const *argv[])
{
   int n;
   int s1,s2;
   int r1,r2;
   printf("Enter the number:\n");
   scanf("%d",&n);
   s1=sqr(n);
   printf("square number: %d \n",s1);
   r1=reverse(n);
   printf("reversed number:%d \n",r1);

   s2=sqr(r1);
   printf("square number: %d \n",s2);

   r2=reverse(s2);
   printf("reversed number:%d \n",r2);
   if(s1==r2)
```

```
    {
        printf("It is a Adam Number");
    }
    else
    {
        printf("Not a Adam Number");

    }
    return 0;
}
```

OUTPUT:

```
Enter the number:
12
square number: 144
reversed number:21
square number: 441
reversed number:144
It is a Adam Number
```

HAPPY NUMBER USING FUNCTIONS

```
#include <stdio.h>
int Hap(int num)
{
    int rem = 0, sum = 0;
    if(num==0){
        return 4;
    }
    while(num > 0)
    {
        rem = num%10;
        sum = sum + (rem*rem);
        num = num/10;
    }
    return sum;
}

int main()
{
    int num ;
    scanf("%d",&num);
    int temp = num;

    while(temp!=1 && temp!=4)
    {
```

```
    temp = Hap(temp);
  }
  if(temp == 1)
    printf("%d is a happy number", num);
  else
    printf("%d is not a happy number", num);

  return 0;
}
```

OUTPUT:

45
45 is not a happy number

ARRAY

ARRAY IS A DATA STRUCTURES OR ARRAY IS THE COLLECTION OF ITEMS OF SAME DATA TYPE