Task 1: a. Generate the following sequence with the recursive approach

1 , 3 , 6 , 10 , 15 , 21 , 28 . . . .

```
#include <iostream>
using namespace std;

void funct(int n, int i,int sum);

int main()
{
	int num,i=1,sum=0;
	cout<<"Enter the number of terms to be displayed: ";
	cin>>num;
	funct(num,i,sum);
}

void funct(int n, int i, int sum)
{
	if(n>0)
	{
		sum+=i;
		cout<<sum<<", ";
		funct(n-1,i+1,sum);
	}

}
```

b. Generate the following sequence with recursive approach

```
               1 , 1 , 2 , 4 , 7 , 11 , 16 , 22 . . . .
```

```
#include <iostream>
using namespace std;

void funct(int n, int i,int sum);

int main()
{
	int num,i=0,sum=1;
	cout<<"Enter the number of terms to be displayed: ";
	cin>>num;
	funct(num,i,sum);
}

void funct(int n, int i, int sum)
{
	if(n>0)
	{
		cout<<sum<<", ";
		sum+=i;

		funct(n-1,i+1,sum);
	}

}
```

Task-2:

a. Write an indirect recursive code for the above task-1 (a,b) part with the same approach as defined in the above sample code of In-Direct Recursion.

```
#include <iostream>
using namespace std;

void funct(int sum);
void funct1(int n,int sum);

int main()
{
	int sum=0;
	funct(sum);
}
void funct(int sum)
{
	int n=1;
	funct1(n,sum);

}

void funct1(int n, int sum)
{
	if(n<10)
	{
		sum+=n;
		cout<<sum<<", ";

		funct1(n+1,sum);
	}
}
```

Task 2b:

```
#include <iostream>
using namespace std;

void funct(int sum);
void funct1(int n,int sum);

int main()
{
	int sum=1;
	funct(sum);
}

void funct(int sum)
{
	int n=0;
	funct1(n,sum);

}

void funct1(int n, int sum)
{
	if(n<10)
	{
```

```
            cout<<sum<<", ";
            sum+=n;

            funct1(n+1,sum);
      }

}
```

Task 3:

Sort The Unsorted Numbers with both tails recursive and Normal recursive approach
Sample Input and Output
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13

```
(Non-Tailed):

#include <iostream>
using namespace std;

void funct(int n, int a[6], int i);
void check(int n, int j, int a[6]);
void print(int n, int a[6]);
int main()
{
      int arr[6]={12,11,13,5,6,7};
      cout<<"Original Array: "<<endl;
      print(5,arr);
      funct(5, arr,0);
      cout<<endl<<"Sorted Array: "<<endl;
      print(5,arr);
}

void funct(int n, int a[6], int i)
{
      if(i<n)
      {
            int j=0;
            check(n-i,j,a);
            funct(n+1-i,a,i+1);
      }

}

void check(int n, int j, int a[6])
{
      int temp;
      if(j<n)
      {
            if(a[j]>a[j+1])
            {
                  temp=a[j];
                  a[j]=a[j+1];
                  a[j+1]=temp;
            }
            check(n,j+1,a);
      }
}

void print(int n, int a[6])
{
      if(n>0)
            print(n-1,a);
      cout<<a[n]<<" ";
}
```

(Tailed):

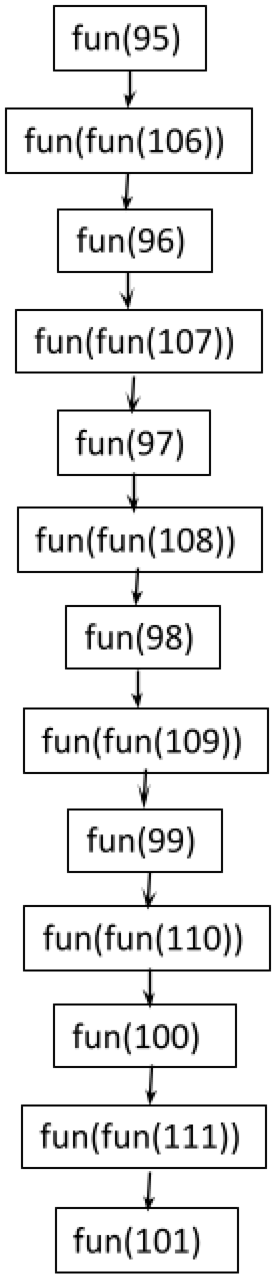
```
#include <iostream>
using namespace std;

void funct(int n, int *a);
void print(int n, int *a);
int main()
{
      int arr[6]={12,11,13,5,6,7};
      cout<<"Original Array: "<<endl;
      print(5,&arr[0]);
      funct(6,&arr[0]);
      cout<<endl<<"Sorted Array: "<<endl;
      print(5,&arr[0]);
}

void funct(int n, int *a)
{
      int temp;
      if(n==1)
      {
            return;
      }
      for (int i=0; i<n-1;i++)
      {
            if(*(a+i)>*(a+i+1))
            {
                  temp=*(a+i);
                  *(a+i)=*(a+i+1);
                  *(a+i+1)=temp;
            }
      }
      funct(n-1,a);
}

void print(int n, int *a)
{
      if(n>0)
            print(n-1,a);
      cout<<*(a+n)<<" ";
}
```

Task 04: Dry run the outputs of the upper code in order to find out how the recursive calls are made



This returns 91.

Task 5a

A. Design the function with recursive approach to find the number of existing destination
path in the above provided sample code link

B. Change the Maze with following configuration .Find the optimal path to reach the destination with recursive approach

```
int maze[N][N] = { { 0, 0, 0, 1 },
                   { 0, 1, 1, 1 },
                   { 0, 1, 1, 0 }}
```

```
#include <iostream>
using namespace std;

bool Solution(int arr[4][4], int arr2[4][4], int r, int d);
bool CheckSafe(int arr[4][4], int r, int d);
void printarr(int arr[4][4]);

int main()
{
	int arr[4][4]={{1,0,0,0},{1,1,0,1},{0,1,0,0},{1,1,1,1}};
	int solved[4][4];
	for(int i=0;i<4;i++)
	{
		for(int j=0;j<4;j++)
		{
			solved[i][j]=0;
		}
	}
	if(Solution(arr,solved,0,0)==true)
		printarr(solved);
	else
		cout<<"No Possible Solutions";
	return 0;
}

bool Solution(int arr[4][4], int arr2[4][4], int r, int d)
{
	if(r==3 && d==3 && arr[r][d]==1)
	{
		arr2[r][d]=1;
		return true;
	}
	if(CheckSafe(arr,r,d)==true)
	{
		arr2[r][d]=1;
		if(Solution(arr,arr2,r+1,d)==true)
		{
			return true;
		}
		if(Solution(arr,arr2,r,d+1))
		{
			return true;
		}
		arr2[r][d]=0;
		return false;
	}
```

```
      return false;
}

bool CheckSafe(int arr[4][4], int r, int d)
{
      if(r<4 && d<4 && arr[r][d]==1)
      {
            return true;
      }
      else
      {
            return false;
      }
}

void printarr(int arr[4][4])
{
      for(int i=0;i<4;i++)
      {
            for(int j=0;j<4;j++)
            {
                  cout<<arr[i][j]<<" ";
            }
            cout<<endl;
      }
}
```

Task 5b:

```
#include <iostream>
using namespace std;

bool Solution(int arr[3][4], int arr2[3][4], int r, int d);
bool CheckSafe(int arr[3][4], int r, int d);
void printarr(int arr[3][4]);

int main()
{
      int arr[3][4]={{1,1,0,1},{0,1,1,1},{0,1,1,1}};
      int solved[3][4];
      for(int i=0;i<3;i++)
      {
            for(int j=0;j<4;j++)
            {
                  solved[i][j]=0;
            }
      }
      if(Solution(arr,solved,0,0)==true)
            printarr(solved);
      else
            cout<<"No Possible Solutions";
      return 0;
}

bool Solution(int arr[3][4], int arr2[3][4], int r, int d)
{
      if(r==2 && d==3 && arr[r][d]==1)
```

```
    {
        arr2[r][d]=1;
        return true;
    }
    if(CheckSafe(arr,r,d)==true)
    {
        arr2[r][d]=1;
        if(Solution(arr,arr2,r+1,d)==true)
        {
            return true;
        }
        if(Solution(arr,arr2,r,d+1))
        {
            return true;
        }
        arr2[r][d]=0;
        return false;
    }
    return false;
}

bool CheckSafe(int arr[3][4], int r, int d)
{
    if(r<3 && d<4 && arr[r][d]==1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void printarr(int arr[3][4])
{
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
        {
            cout<<arr[i][j]<<" ";
        }
        cout<<endl;
    }
}
```