

A ■ P ■ I ■ I ■ T

ASIA PACIFIC INSTITUTE OF
INFORMATION TECHNOLOGY

RECURSION

Introduction

RECURSION

This is where a function repeatedly calls itself to perform calculations. Typical applications are games and Sorting trees and lists.

Consider the calculation of $6!$ (6 factorial) i . e .

$$6! = 6 * 5 * 4 * 3 * 2 * 1$$

$$6! = 6 * 5!$$

$$6! = 6 * (6 - 1) !$$

$$n! = n * (n - 1) !$$

$$= n * (n - 1) * (n - 2) * * 1$$

Properties

- There must be some situation in which a recursive function does not invoke itself; That means every recursive function must have a **base case**
- The function invoked always returns to its invoker.
- When a function invokes itself, the values used in testing for the base case must change

```
/* An example for demonstrating recursion */  
#include <stdio.h>  
long int factorial( long int ); /* ANSI function  
                                prototype */  
  
long int factorial( long int n )  
{  
    long int result;  
    if( n == 0L )  
        result = 1L;  
    else  
        result = n * factorial( n - 1L );  
    return ( result );  
}
```

Recursion and Function
Prototype

```
main( ) {  
    int j;  
    for( j = 0; j < 11; ++j )  
        printf("%2d! = %ld\n",  
                factorial( (long) j) );  
}
```

Example :

$!5 = 5 * !5-1$

$4 * !3$

$3 * !2$

$2 * !1$

$1 * !0$

Results 120 ←

returns 24 ←

returns 6 ←

returns 2 ←

returns 1 —

REVERSING A STRING

```
#include<stdio.h>

void StackChar()
{
    char c;
    scanf("%c",&c);
    if(c!='\n')
        { StackChar();
          printf("%c",c);}
}

void main(){
    printf("Enter a string to be reversed;");
    StackChar(); printf("\n");
}
```



```
#include<stdio.h>

void StackChar(int level)
{
    char c;
    ++level;
    scanf("%c",&c);
    printf("Reading at level %d\n",level);
    if(c!='\n') { StackChar(level);
                printf("Printing at level %d : %c\n",level,c);}
}

void main(){
    printf("Enter a string to be reversed;");
    StackChar(); printf("\n"); printf("\n");
}
```

Multiplication of Natural Numbers

The product $a*b$, where a and b are +ve integers, may be defined as a added to itself b times. An equivalent recursive definition is

$$a*b = a \quad \text{if } b == 1$$

$$a*b = a*(b-1) + a \quad \text{if } b > 1$$

Example : To evaluate $6*3$ by the definition, we first evaluate $6*2$ and then add 6. To evaluate $6*2$, we first evaluate $6*1$ and then add 6.

$$\text{Thus } 6*3 = 6*2 + 6 =$$

$$6*1 + 6 + 6 =$$

$$6 + 6 + 6 = 18$$

Calculating Powers Recursively

```
#include<stdio.h>
```

```
float power(float x,int n) {
```

```
    if(n>0) return(x*power(x,(n-1)));  
    return(1.0); }
```

```
void main() {
```

```
    float x;
```

```
    int  n;
```

```
    printf("Enter x and n :");
```

```
    scanf("%f %d", &x,&n);
```

```
    printf("%f ^ %d \n",x,n,power(x,n));
```

```
}
```

```
#include<stdio.h>

float power(float x,int n, int level) {
    float tempPower;
    ++level;
    if(n > 0)
        { tempPower = x*power(x,n-1,level);
          printf(“Level %d About to return value %f \n”,
                                                         level,tempPower);

          return(tempPower);
        }
    printf(“Bottom of recursion reached at level %d\n”,level);
    return(1.0);
}
```

```
Void main()
{ float x; int n;
  printf("Enter x and n :");
  scanf("%f %d",&x,&n);
  printf("%f ^ %d = %f \n",x,n,power(x,n));
}
```

A run to calculate 2^3 produces :

Enter x and n: 2 3

Bottom of recursion reached at level 4

Level 3 About to return the value 2.000000

Level 2 About to return the value 4.000000

Level 1 About to return the value 8.000000

$2.000000^3 = 8.000000$

FIBONACCI SERIES

The Fibonacci sequence is the sequence of integers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3542248, 5713007, 9254255, 14930352, 24214709, 39186360, 63405337, 102573236, 166778125, 269271571, 436049686, 705307861, 1141772136, 1856072835, 3007804971, 4863882806, 7871687777, 12635568582, 20517254359, 33154682941, 53671241523, 86825924464, 139583136987, 226409106451, 366032143538, 592441250000, 958493493538, 1551413215039, 2511886465581, 4063399680620, 6575286146201, 10638685826821, 17213962002021, 27851654121842, 45065616123863, 72917579245884, 117979235269747, 190915627015631, 308884862285478, 500047365619062, 809046191703041, 1309060545216355, 2121370304461175, 3431436854685125, 5552808344680500, 8984369578125625, 14536180482707125, 23520549660414150, 38056730161343075, 61577280144157225, 100635016914097375, 162207197058254600, 262882213972451975, 425089410630649175, 687296617688901175, 1112378818661353150, 1809675436350254325, 2922054255011607475, 4731729691361861600, 7653783946373469075, 12385513637735270675, 20039302829108739750, 32425016466844010425, 52464320104579250175, 84903636570423260600, 137367956675002460775, 222271593245425710975, 359639550015428171650, 581906506690430572625, 941546056705856283575, 1523452563406284455200, 2464958619996714627775, 3988411183403000083000, 6453369803399714700775, 10441880986802714783775, 16895250790201714408000, 27337120593594429191775, 44232371579797143874775, 71569622370001558066500, 115806742963598687258275, 187376365333600831324775, 303183088297198418581275, 490559831260799305840000, 793736206657998137164775, 1283919287954197442906050, 2077655494612196580066825, 3361574782566393717172875, 5439230277178590297239600, 8800805069740783914406475, 14239035346919374111639275, 23039830416660164308878900, 37278865766399538420518175, 60318896183059702532157375, 97558761949459240652676250, 157877658132518942174834000, 255436420111578144606991775, 413314078244097086779668750, 668751498355675228986660500, 1082065568467272315666329250, 1750817066822869544646000000, 2832882665180141773632668750, 4583699733647011289598988750, 7416562401567153063231657500, 11999442134847394352829646250, 19416031836414547416061335000, 31415493971261741769301031250, 50831525807709036122130676250, 82246917939180777538192007500, 133078443746889813660322683750, 215320361686070590808514706250, 348398809625259408348837388750, 563719171311330000011352095000, 912117980936589418370189483750, 1475837152247919419181541579000, 2387956133184508837551730663750, 3863773285432428255822912242500, 6251729418617347673374642906250, 10115502604050266929196555150000, 16367232022667614602568288056250, 26482734626717881531764843206250, 42849966649385496134333131262500, 69332699676003380736097974468750, 112182666325391266867831815725000, 181515365991394647603829790187500, 293698032316785928371661605912500, 475213408308180575975489596100000, 768911440624966504347151201812500, 1243124848933146480322640807912500, 2011036289558127056297840399012500, 3254161138491273536619481190925000, 5265197428049400592917321589937500, 8519358566540673629536802780862500, 13784555994589874226454124370800000, 22303914562630474829371446150737500, 36088470557220348455825570521600000, 58392385120850823085196916672337500, 94480855678071171541022487193937500, 152873240798922019626219403866275000, 247354126466993192667241320538612500, 400227367265915211788263724404887500, 64758151376490840445550504494350000, 1047808881030823616243768769348387500, 1695390394795732020700273814291887500, 2743199275826555636943942583640275000, 4438589670622287657644216397932162500, 718178894644884227854816098162405000, 11619378617071129936192377379556212500, 18791167563519972204740538361180262500, 30410546180591102140932915740736475000, 49201713744111074345673454101916737500, 79612260924702176486606370042653212500, 12881397466881327083257982414457000000, 20842623559351534731918619418718312500, 33724021026232861815176501833175562500, 54566418485584396547094421251993875000, 88280441511816928352271023085169437500, 142846859997399825169365444337163312500, 231127271509216753521459467422157187500, 373974131499033681873625490507326612500, 605101403008250435395084957929483812500, 978075534497284117268710448436800425000, 1583176937505534552663795405366284237500, 2558278371992784669958875853295768062500, 4141454309498319222622571258732052300000, 6699732681491103882581447112027820362500, 10841207001489423105204018370759872662500, 1754093968298052698778546548278769302500, 28382146684469949093009483853547565687500, 45923086367450476080794949336335258712500, 74263026051910425173780433190122824337500, 12018605241936089525457538252645808305000, 19444907847127132123835581571658090737500, 31463513089063221651293119824293901140000, 50908420936190353775128701395951991875000, 82371934025253575426421821219645893012500, 133280354961443929197615522615637884862500, 21565228598669750462403734383558377787500, 34893264094814142405045886645521967087500, 56458492693483935324807618927080345875000, 91351756788298077729853503312602312962500, 147800249481712212054661122239683658837500, 239152006270010289784514630552286071812500, 386952255751722501814367752791888390312500, 62610426202173271486898138334417446212500, 101305641777345521668334913613606285237500, 16391606807951879315523305194802373145000, 265221709856864314823621165561630016687500, 429137777936383008078854217509653748137500, 694363787793247322902475383071283764812500, 1119591565729630330981129590580937512962500, 18139553535229133538835049736522212775000, 293354691925254368486463456423315879062500, 474750227277545703874813953788538006812500, 76810482420279703236127741021185388587500, 124285505148034273623609136400039189837500, 201095987568313973960090877421224578437500, 325381492716348247583700013821363768312500, 526477480284662221543790891242588346787500, 851858972990910469127490905063852115137500, 137833645327557271667128179630643946187500, 222919542626648318681477270154929157937500, 360753187954205590348605449785573069687500, 583672730580853908929982720040502227637500, 944425918535059499278588169826075297312500, 152809864911581340820857088986657752487500, 247252456765087231748855360990708000237500, 400062321676668572569712449977365752737500, 647314778441755804318567810968073753012500, 1047567100118424376888280260945339505762500, 1694881878560180181206848071913413258812500, 2742448978678604558095128332858752764562500, 443733085723878473930197640477216602337500, 717977883590896492049710473768591878787500, 1161710969314774965979898114245808481137500, 187968885280567145802960858801439036012500, 30414008221204464240095066922602088412500, 49210896749261183820391152802745992012500, 79624904970465648060486219725348080437500, 12883579171972683188087737252809407245000, 20846069669019248004136359225344215287500, 337296488409919311902250804978790225312500, 545647180102611791783614397231232378187500, 882943676512531103785865202210022603512500, 1428590856615142895669479599441254981687500, 231153803671775468745309399665148736012500, 3739689193332896791138959206092740041687500, 6051227229948041478592053202744227401812500, 978041642328093826973001240883696744337500, 1583164365322907974832206561158119485587500, 2558288727317712122691411881432542225812500, 4141453092640620097523618442590661970312500, 669973181995853227041503032392310419587500, 1084120583260915236793864876651376616812500, 1754093365256768463835367909043687036312500, 283821494851768370062923278569506365312500, 459230831377445216446460069473875068937500, 742630208003121062830023360478193872687500, 1201860439380566279276483429951989939187500, 1944490647383687341106506790430183816687500, 314635128676425362038299021038217375612500, 509084203414794096148949700081235757262500, 823719331091220458187248721119453132812500, 1332803534506014554336198421200688890312500, 215652280569723465252544714231992402312500, 348932653689444911071164586352037713562500, 564584934259168376323709290584030115812500, 913517565828886287394873876936067828312500, 1478002400088054663718583167520107940812500, 2391524865347243040013286365104138053312500, 3869527330606431426731869532624245965812500, 626104969625367446674515589772838401937500, 1013057498690014589347692543035262998187500, 1639164968315382036022208132808101400687500, 265221746690540162536989067584336439812500, 429137993322078371141710880887856580062500, 694364239953616572278629948472186620312500, 1119590479875694943416340836359343192812500, 1813953009829311515694960784831529813312500, 2933545479654906459111301621190872925812500, 4747502949480490974806262406022402738312500, 768104589896098294961252481205480547312500, 1242856579844147512441878722807720820812500, 2010959059720245807403131204039201368312500, 325381661944039331984426292684692218312500, 526477417916053912724739465088612354562500, 851858984832083244709165757673304572812500, 1378336459748136467433895222761916927312500, 2229195429668283692158634689845529280312500, 3607531899488430151592530912607446095812500, 5836727329308603843751165592452975308312500, 944425919912875769534379650506042140312500, 1528098649832935148703659301761339670812500, 2472524569753000317872818861016860201312500, 4000623239573146469366478162778200016812500, 6473147809393292287239296964539360231312500, 10475671507594750556605995027317560451812500, 16948818795188042843845291991856920676812500, 274244896927894933994512870191744811312500, 443733083903909439550572790160314017812500, 717977880885924018044985700352058829312500, 1201860450765948456595558490543872840812500, 1944490698685126941191087681086965053312500, 3146351397445272397786646171630837865812500, 5090842145365418338977733852717802918312500, 823719349328556415496332195424863102312500, 1332803597204603654393010390696726313812500, 2156522845124749503584043581221509425812500, 3489326592944895322977153772742371538312500, 564584934082504114086826396326315265312500, 913517568870518695684937415378402376812500, 1478002406840664511740047611704714588312500, 239152487676081033012915780222550069312500, 386952735154095614801976800374637180812500, 626104970942110196674093610526724291312500, 1013057499822124778273197420681356402812500, 1639164969581244996748216231203080415312500, 2652217448341390815027335441724160427812500, 4291379918101536633920445642245240439312500, 6943642387881682451791555832966321554312500, 11195904876682169239702666023497642669312500, 18139530176218344960493768128705503784312500, 29335454974474294400299869942193145909312500, 47475029672676142579205971856680757034312500, 76810459370878000758917073761888268184312500, 124285658708892622658028095617095979334312500, 201095906696910105365939116671393090484312500, 325381663678924687115050137723480201634312500, 526477419650939268804161158775577312884312500, 851858984526953850593272178927664424384312500, 1378336461407000031706383198080236535884312500, 2229195431327145810087493408601812647384312500, 3607531901147291628988603609122683760384312500, 5836727330967437446879713809643554875384312500, 9444259200787583264870824000164335980384312500, 15280986509588134081941934210687697095384312500, 24725245708789592269733036306199308200384312500, 40006232407590050469644138316406409315384312500, 64731478105791508648355240221615120430384312500, 104756715163806086517066360843230231545384312500, 169488188039723585165421621964845332695384312500, 274244897015738167052532642917053443845384312500, 443733084013752748841643663069164555345384312500, 717977881013767330730754683121270666845384312500, 1201860451677813512401864893273341778345384312500, 1944490699597959330692975103816422889845384312500, 3146351408358005148688085314337294000345384312500, 5090842155178150965579195524858265111845384312500, 8237193501998296781570305726389136226345384312500, 13328036071780342562681415937900271341345384312500, 21565228470981800750592518043108082456345384312500, 34893265949183258938503620058316793571345384312500, 56458493427984716717414722063525504721345384312500, 91351756906786174897925824118734215871345384312500, 147800240787800756698036844229955427021345384312500, 239152487779815338577147865282033538171345384312500, 386952735275829920466258885334142649421345384312500, 626104971063844502255369905386230760921

FIBONACCI SERIES

$$\text{fib}(n) = \text{fib}(n-2) + \text{fib}(n-1) \quad \text{if } n > 2$$

For an example

$$\begin{aligned}\text{fib}(6) &= \text{fib}(4) + \text{fib}(5) = \text{fib}(2) + \text{fib}(3) + \text{fib}(5) = \\ &0 + 1 + \text{fib}(3) + \text{fib}(5) = 1 + \text{fib}(1) + \text{fib}(2) + \text{fib}(5) = \\ &1 + 1 + \text{fib}(0) + \text{fib}(1) + \text{fib}(5) = \\ &1 + 1 + 0 + 1 + \text{fib}(3) + \text{fib}(4) = \\ &3 + \text{fib}(1) + \text{fib}(2) + \text{fib}(4) = \\ &3 + 1 + \text{fib}(0) + \text{fib}(1) + \text{fib}(4) = \\ &4 + 0 + 1 + \text{fib}(2) + \text{fib}(3) = \\ &5 + \text{fib}(0) + \text{fib}(1) + \text{fib}(3) = \\ &5 + 0 + 1 + \text{fib}(1) + \text{fib}(2) = 6 + 1 + \text{fib}(0) + \text{fib}(1) \\ &7 + 0 + 1 = 8\end{aligned}$$

FIBONACCI SERIES

```
fib(n)
int n;
{
    int x,y;
    if(n <= 1)    return(n);
    x = fib(n-1);
    y = fib(n-2);
    return(x+y);
}
```

Example : fib(5)

1 1 2 3 5

TOWER OF HANOI

This is a ‘true’ story which dates back to very ancient times...

Somewhere in the far east, there exists a Monastery with a pile of sixty four (!64) disks of increasing size made of pure gold, stacked one on each other, with the largest on the bottom. Each disk is pierced with a hole and they rest in a single pile on a pin(peg) of purest diamond!

[For more complete, lucid details, including location maps and access means, refer “Mathematical Recursions and essays” by W.W. Rouse Ball]

TOWER OF HANOI

The monks played the following game at their leisure ..

Move the golden disks from the diamond peg to another peg, according to the following rules:

- * Only one disk to be moved at a time
- * A large disk could never be placed on top of a smaller one
- * The original pile could be moved from one position (Say A) to another position(Say C) using a single additional temporary position (Say B) about a peg.

TOWER OF HANOI

Move from peg A to peg C can be represented as $A \rightarrow C$

‘n’ represents number of disks

Analysis of “ordinary” situations:

$n = 1$: $A \rightarrow C$ [1 Move]

$n = 2$: $A \rightarrow B, A \rightarrow C, B \rightarrow C$ [3 Moves]

$n = 3$: $A \rightarrow C, A \rightarrow B, B \rightarrow A, A \rightarrow C, B \rightarrow C, A \rightarrow B, B \rightarrow C$ [7 Moves]

TOWER OF HANOI

The Problem :

Using the concept of Recursion, it is required to move 'n' disks from starting peg 'A' to ending peg 'C', using an auxiliary peg B.

SOLUTION :

- Move the top $n-1$ disks from peg A to peg B.
- Move the top disk from peg A to peg C
- Move the top $n-1$ disks from peg B to peg C

TOWER OF HANOI

Mathematical equation for this Recursive problem :

$$M(n) = \begin{cases} 1, & n=1 \\ 2.M(n-1) + 1, & n>1 \end{cases}$$

Since moving a pile of n disks involves moving a pile of n-1 disks, then moving 1 and then moving the pile of n-1 disks.

This recurrence relation generates

| | | | | | | | |
|------|---|-----------|---|---|----|----|------|
| n | = | 1 | 2 | 3 | 4 | 5 | 6... |
| M(n) | = | 1 | 3 | 7 | 15 | 31 | 63 |
| M(n) | = | $2^n - 1$ | | | | | |

TOWER OF HANOI

Move all disks from A to C using B as auxiliary peg.

- 1) If $n = 1$ move the disk from A to C
- 2) Move top $n-1$ disks from A to B using C as auxiliary.
- 3) Move the n th disk from A to C.
- 4) Move $n-1$ disks from B to C using as auxiliary.

TOWER OF HANOI

Now revert to the original story

Turning point

For $n = 64$ disks

$M(n) = 1.84 \text{ E}+19$ moves

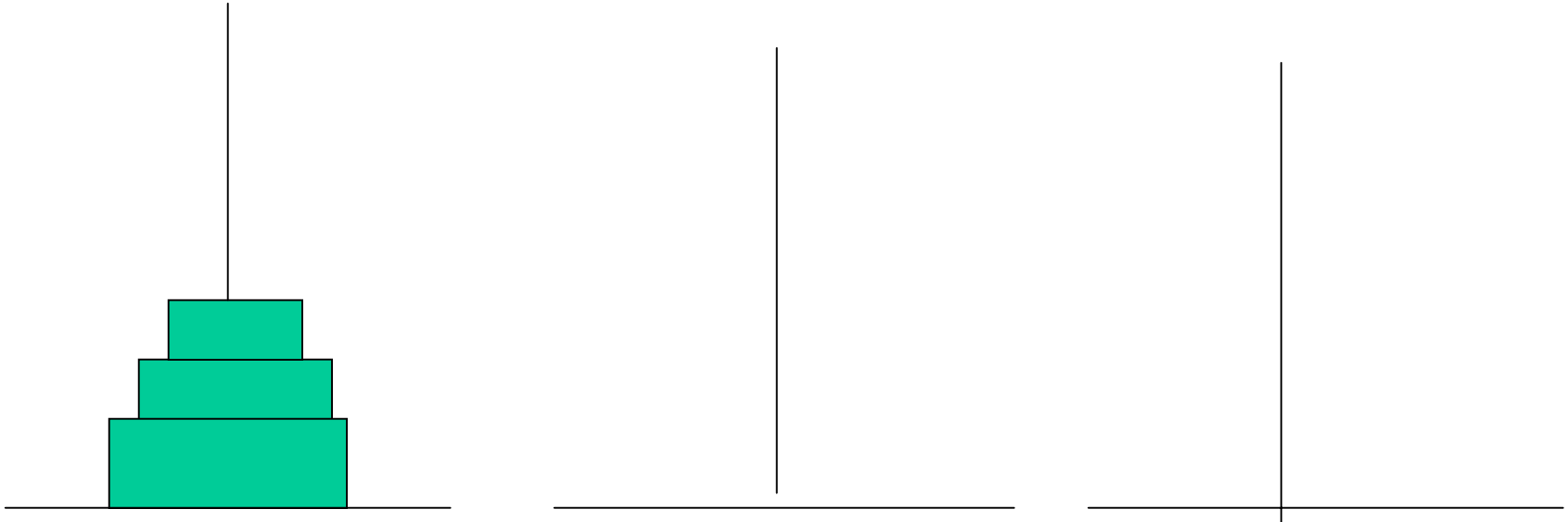
If the monks worked at the rate of one move per second, then this would require about **585 billion years !!!**

A present day computer operating at one Million moves per second takes about **585 thousand years**

__Finally

TOWER OF HANOI

When this “Towers of Hanoi” is completely moved from one position to another, END OF THE WORLD will come !!!!!



TOWER OF HANOI

```
main()
```

```
{ int n;
```

```
    scanf("%d",&n);
```

```
    towers(n,'A','B','C');
```

```
}
```

```
towers(n,frompeg,topeg,auxpeg)
```

```
int n; char frompeg,topeg,auxpeg;
```

```
{ if(n == 1)
```

```
    { printf("\n %s%c%s%c", "Move disk1 from
```

```
        peg",frompeg,"to peg",topeg,);
```

```
    return; }
```

```
towers(n-1,frompeg,auxpeg,topeg);  
printf("\n %s%c%s%c", "Move disk", n, "from  
Peg", frompeg, "to peg", topeg);  
towers(n-1,auxpeg,topeg,frompeg);  
  
}
```

EXERCISE

Write a recursive function `ODD_TIMER(N)` that finds the product of first N odd numbers. The N th odd number is given by the formula $2*N-1$. i.e. The 3rd odd number is $2*3-1 = 5$.

E.g. :-

$$\begin{aligned}\text{ODD_TIMER}(4) &= 1*3*5*7 \\ &= 105\end{aligned}$$

```
#include <stdio.h>

/*Procedure To Calculate Odd Numbers */
ODD_TIMER(N)
int N;
{
    int x,y;
    if (N <= 0)
        return(1);
    x=N-1;
    y=ODD_TIMER(x);
    return((2*N-1) * y);
}
```

```
/*Procedure To Check If N>0 Or To Display  
The ODD TIMER Of N*/  
checking(N)  
int N;  
{ int x;  
  if (N > 0)  
    x = ODD_TIMER(N);  
  else  
    x = 0;  
  printf("\n The ODD_TIMER of (%d) = %d  
\n",N,x);  
}
```

```
/*Main Program To Display The Message And  
To Read The Input*/  
  
/*And Also To Display If Invalid Input  
Have Entered*/  
  
main()  
{ int N,x;  
  printf(" Please Enter N Odd Numbers :  
");  
  scanf("%d", &N);  
  if (N >= 0)  
    checking(N);  
  else  
    printf("\n Invalid Input Have Entered  
\n");}
```

THE END OF III LESSON