```
// do-while loop


/* ------------------------------------  Qu 1: Diff b2n do-while and while/for
  i=1;   // init...
  do{
   printf("Calcutta University\n");
   i++;
  }while(i>2);

  // ------------------------------------
  i=1;
  while(i>2)
  {
     printf("Calcutta University\n");
     i++;
  }
 */


 // ------------------------------------   Qu 2: applications of Do-While Loop
#include <stdio.h>
int main()
{
   int i, n;          // Consider i for each line, j for each Star and k for each
Space

   do{
     printf("Enter a positive number:\t");
     scanf("%d",&n);   // let n=-5     -19     18
   }while(n<0);

   printf("We have recvd a +ve no %d", n);

   return 0;
}
```

```
#include <stdio.h>
void hello()          // function definition : function body
{
   printf("HI How are you");
}

int main()
{
   hello();    // function calling: calls the hello funtion    //hello is a function
bcz it considers ()      while()   xyz()   if()   ...;
   return 0;
}
```

```
#include <stdio.h>
void hello();   // function prototype:
int main()
{
   hello();    // function calling: calls the hello funtion    //hello is a function
bcz it considers ()      while()   xyz()   if()   ...;
   return 0;
}

void hello()          // function definition : function body
{
   printf("HI How are you");
}
```

```
/*pointer*/

#include <stdio.h>
int main()
{
   int a=2, *p; // p is a pointer that can hold address of a variable
   p=&a;
   printf("value of a is %d", a);   //2
   printf("value of a is %d", *p);  //2
   printf("address of a is %u", &a);   //100
   printf("address of p is %u", &p);    //200



   int a=2, *p, **q, ***r, ****s;   // p, q, r, s are dangling pointers
   p=&a;
   q=&p;
   r=&q;
   s=&r;

  //  variable [value] memory address
  //       a  [2]      100
  //      p  [100]     200
  //      q  [200]     300
  //      r  [300]     400
  //      s  [400]     500

  a  2   &a 100
  p 100  &p 200    *p 2
  q  200 &q 300    *q 100  **q 2
  r  300 &r 300    *r 200  **r 100   ***r 2
  s  400 &s 500    *s 400  **s 300   ***s 200  ****s 2

  //array with pointer
  //note "array name" denotes the "address of its first cell"

  int a[30]={1,2,3,4,5}, *pa;
```

```
  pa=&a[0];  //pa is a pointer that points to an array variable   "pa=a;"

  for(i=0;i<5;i++)
  {
     printf("%d", a[i]);
     printf("%d", *(pa+i));
  }

  e.g.:   0  1    2    3    4   array index
          100  102   104   106  108 array memory address
       a  1   2    3    4    5   array values
       pa 100

       pa+0  100   *(pa+0)  1
       pa+1  102   *(pa+1)  2
       ...
       pa+4  108   *(pa+4)  5

    return 0;
}
```

```
// adding 2 numbers

#include <stdio.h>
int main()
{
   int a,b,c;
   scanf("%d %d",&a, &b);
   c=a+b;
   printf("Sum is %5d\n",c);
   return 0;
}

// adding 2 numbers  using 1000 times (Looping)

#include <stdio.h>
int main()
{
   int a,b,c;
   for(i=0; i<100; i++)
   {
     scanf("%d %d",&a, &b);
     c=a+b;
     printf("Sum is %5d\n",c);
   }
   return 0;
}



// adding 2 numbers  using functions without arguments
#include <stdio.h>

void add()              //function definition is done:
{
   int a, b, c;
   scanf("%d %d",&a, &b);
   c=a+b;
```

```
    printf("Sum is %5d\n",c);
}

int main()
{
    add();    //function calling
    return 0;
}



// adding 2 numbers  using functions with arguments
#include <stdio.h>

void add(int x, int y)              //function definition is done:
{
    c=x+y;
    printf("Sum is %5d\n",c);
}

int main()
{
    int a,b;
    scanf("%d %d",&a, &b);
    add(a, b);    //function calling using arguments but without any return
value
    return 0;
}



// adding 2 numbers  using functions with arguments and return value
#include <stdio.h>

int add(int x, int y)               //function definition is done: x& y are called
formal arguments
{
    return(x+y);
```

```
}

int main()
{
    int a,b;
    scanf("%d %d",&a, &b);
    c= add(a, b);          //function calling using "actual" arguments   //c=5
    printf("Sum is %5d\n",c);
    return 0;
}
```