Think about a situation where we do not know how many elements are required? How many numbers should be given as input? Will an array suffice? Let us say, we are declare an integer array as follows

```
int num[4];
```

we have declared that only 4 members/ numbers are possible to be placed in this array isnt it so? Now, if the user wants more numbers to be entered? Then this array size will be inadequate. But if we declare an array as given in the code below, and if the user enters only 3 numbers, then it would result in a non utilization of memory.

```
int num[100];
```

To overcome such a disadvantage with array, we can dynamically allocate memory at run time.

Let us visualize a scenario where each person points to the next person beside her/ him. Thus, if we get access to the first person, then the first person can point us to the second person and the second person can point to the third person and so on. We have briefly encountered pointers earlier. A pointer is a variable which has the address of another variable.

Let us assume each person in the scenario mentioned above has a number and the address of the person next to her/ him. Have we studied anything earlier which we can use to address this situation? We have structures, which can include multiple data types as a single unit. Let's look at the code given below.

```
3   struct node
4   {
5       int num;
6       struct node *nextptr;
7   }*n1;
```

A structure called as 'node' having two members, 'num' and a pointer named 'nextptr' to point to a 'node' structure (remember there can be many structures) . '*n1', is another pointer to the structure. Now if the user says he wants to enter 5 numbers, then n = 5. It means that the user will enter 5 numbers. In our program, the first number should point to the second number and so on until we reach the fifth number. Let us look at the code given below:

```
struct node *nextNode, *tmp;
int num, i;
n1 = (struct node *)malloc(sizeof(struct node));

if(n1 == NULL)
{
    printf(" Memory can not be allocated.");
}
else
{


    printf(" Input data for node 1 : ");
    scanf("%d", &num);
    n1->num = num;
    n1->nextptr = NULL;
    tmp = n1;
    for(i=2; i<=n; i++)
    {
        nextNode = (struct node *)malloc(sizeof(struct node));

        if(nextNode == NULL)
        {
            printf(" Memory can not be allocated.");
            break;
        }
        else
        {
            printf(" Input data for node %d : ", i);
            scanf(" %d", &num);

            nextNode->num = num;
            nextNode->nextptr = NULL;

            tmp->nextptr = nextNode;
            tmp= nextNode;
        }
```

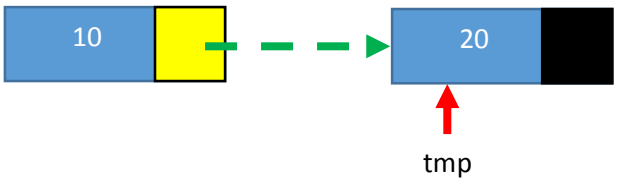
Using malloc to create memory for entering first number. Remember that each number is entered into a structure called as 'node' . n1 is a pointer that points to the base address (first address) of the total memory allocated to this structure

The value provided by the user (say 10) present in the variable 'num' is assigned to the structure member 'num' using the n1 pointer.

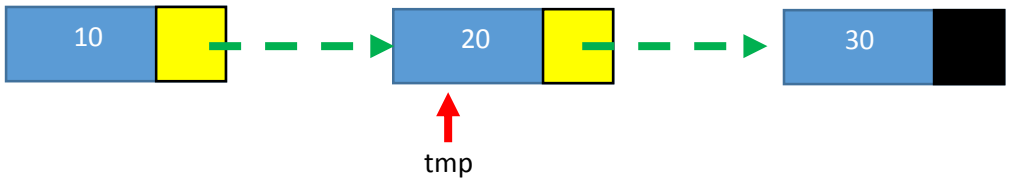
Please note, n = 5. The loop will run until i < = 5

Memory for another structure 'node' is allocated to store the next number

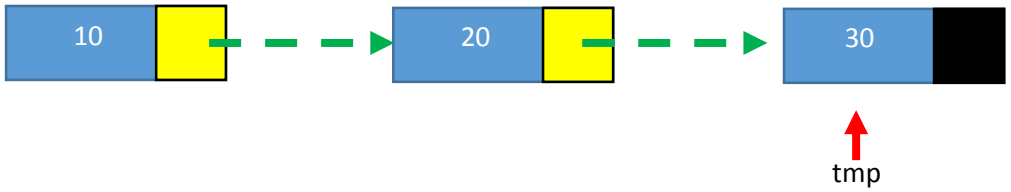
The value provided by user, say 20



The last line of the code, says that let the pointer named ‘tmp’ point towards the latest node.
Thus when the next node is created, tmp -> nextptr will point to the next node



tmp = nextnode



Thus the memory gets allocated until the loop stops.