Functions available in C for dynamic memory management

Function	Description
void *calloc(int num, int size)	Allocates an array of num elements each of which size in bytes will be size.
void free(void *address)	Releases a block of memory block specified by address.
void *malloc(int num)	Allocates an array of num bytes and leave them initialized.
void *realloc(void *address, int newsize)	Re-allocates memory extending it up to new size.

One needs to state **#include <stdlib.h>** at the top of the program.

malloc Function

- The name **malloc** stands for "memory allocation". *The function malloc() reserves a block of specified memory size and returns starting address as a void pointer which we cast into pointer of any form.*

- ***Syntax***: ptr = (cast-type *) malloc(size in byte);

Here, ptr is pointer of cast-type. The malloc() function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.

malloc Function

Syntax: ptr = (cast-type *) malloc(size in byte);

Example:

-
  ```
  int *ptr;
  ptr=(int *) malloc(100);
  ```
  //Allocates 100 Bytes. If size of int is 2Bytes, then 50 int space is allocatead and return fisrt location.

-
  ```
  int *ptr;
  ptr=(int *) malloc(100*sizeof(int));
  ```
  //Allocates either 200 or 400 Bytes according to size of int 2 or 4 bytes respectively and the pointer points to the address of first byte of memory.

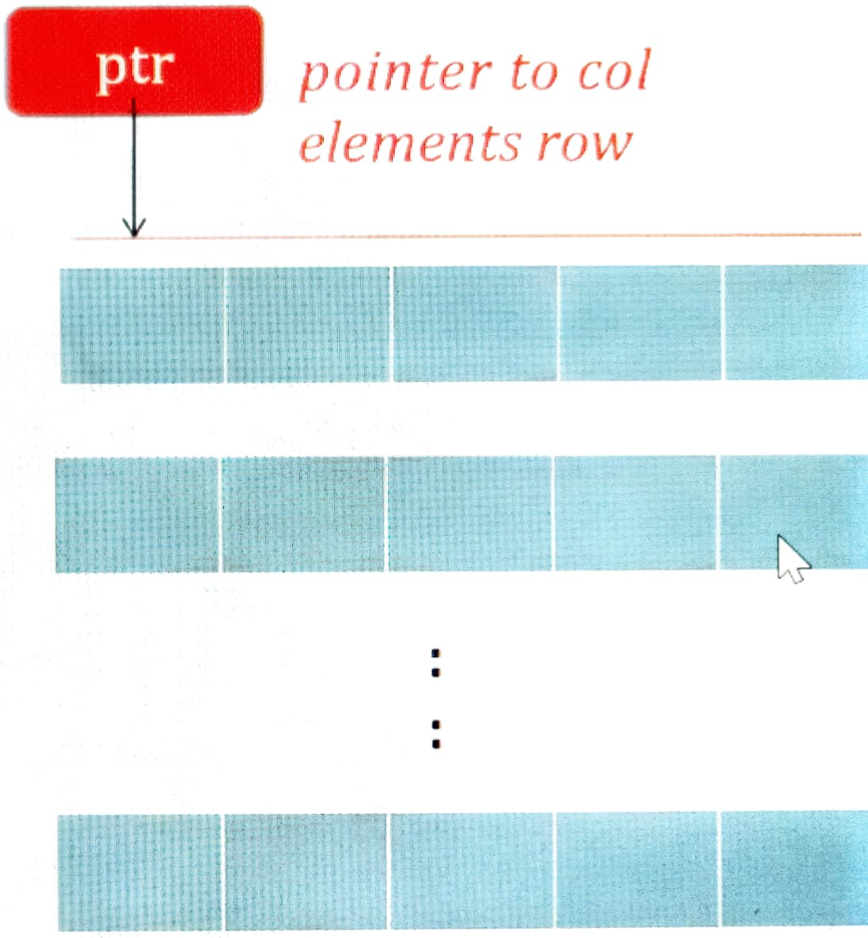
-
  ```
  int *ptr;
  ptr=(int *) malloc(100*sizeof(char));
  ```
  //Allocate either 100 Bytes and pointer points to the address of first byte of memory.

Allocate n elements dynamically to a pointer variable, asign data and find sum of all n elements using pointer variable

```
int main()
{
   int n, i, *ptr, sum=0;
   printf("Enter number of elements: ");
   scanf("%d",&n);
   ptr=(int*)malloc(n*sizeof(int));        //memory allocated using malloc
   printf("Enter elements of array: ");
   for(i=0; i<n; ++i)
   {
      scanf("%d", ptr+i);
      sum +=  *(ptr+i);     // ptr[i];
   }
   printf("Sum=%d",sum);
   free(ptr);
   return 0;
}
```

Allocate & assign dynamically 2D Array ([n][col]) where column size is fixed to col to a pointer and find sum of all elements.

```
#define col 5
int main()
{
int n, i, sum =0 ;
int (*a)[col];              //pointer to 5 elements row
printf("Enter number of rows: ");
scanf("%d", &n);
a = (int (*)[col]) malloc(n * col * sizeof(int));
for(i = 0; i < n; i++)
   for (j=0; j <col; ++j)
   {
     scanf("%d", &a[i][j]);
     sum+= a[i][j];          //  *(*(a+i) + j)
   }
 return 0;
}
```



calloc Function

- Calloc stands for "contiguous allocation". The only difference between malloc() and calloc() is that, **malloc() allocates single block of memory** whereas **calloc() allocates multiple blocks of memory each of same size** and sets all bytes to **zero**.

- ***Syntax***: (cast-type *) calloc(n, element-size);

  Here, ptr is pointer of cast-type. The calloc() function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.

- **Example**:

```
float *ptr;
ptr=(float *) calloc(25,sizeof(float));
```

  This statement allocates contiguous space in memory for an array of 25 elements each of size of float, i.e, 4 bytes.

Find the Largest Element and store it in the 0th position

```
int main()
{
   int i=0,n;
   float *data;
   printf("Enter total number of elements(1 to 100): ");
   scanf("%d",&n);
   data=(float *)calloc(n, sizeof(float));  /* Allocates memory */
   printf("\nEnter Number %d: ", i+1);
   for(i=1; i<n; ++i)                    /* Stores number entered by user. */
        scanf("%f", data+i);             // &data[i]
    for(i=1; i<n; ++i)
        if(*data < *(data+i))
            *data = *(data+i);
   printf("Largest element = %.2f", *data);
   free (data);
   return 0;
}
```