**Dynamic Memory Allocation :**

**Used When we want to allocate memory during run time.**

**int marks[10]; // fixed size and fixed address … No change in Memory address.**

 **// fixed size. ( no change in size possible**

**We have to use <stdlib.h> hadder file for dynamic memory allocation.**

**It has 4 functions.**

1. **malloc()**
2. **calloc()**
3. **free()**
4. **realloc()**

**malloc()
memory allocation**

**allocate the one memory block given by user. // eg. Reserves 20bytes of block**

**calloc()**

**creates number of blocks. // uses for arrays**

**free()**

**used to free the space after using malloc() or alloc()**

**realloc()**

**if used malloc() or alloc() and need to modified memory block size realloc()**

 **\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**malloc()**

**creates the memory block according to given size ().**

**malloc() function Also returns the address , which points the address of the first byte in that specific block.**

**Syntax :**

**void \* maclloc(size in byte );**

**as it has void pointer as return type it can return Any type of data : int , string, char.**

**ptr = (cast\_type\*) malloc( size in byte);**

**ptr = (int\*)malloc(10);**

**you must cast the pointer according to type of data eg. Here.. (int \*)**

**here, ptr will be int type.**

**int – 2 byte**

**it contains garbage value. And here it can hold 5 int values if one int requires 2 bytes**

**in case because of the some problem if memory is not allocated by malloc() function than It will return null pointer.**

**#include<stdio.h>**

**#include<conio.h>**

**#include<stdlib.h>**

**main()**

**{**

**int n, \*ptr, sum =0, i, \*p;**

**printf(“Enter the size of array”);**

**scanf(“%d”,&n);**

**ptr = (int\*)malloc(n\* sizeof(int));**

**// ptr will point the first byte of the memory block.**

**// now we can use null pointer to see block is created or not.**

**if( ptr == NULL)**

**{**

**printf(“Error : out of Memory”);**

**exit(0);**

**}**

**p = ptr;**

**// right now both have the same address.**

**printf(“Enter the elements in Array”);**

**for( i= 1; i<=n; i++)**

**{**

**scanf(“%d”,ptr);**

**sum = sum + \*ptr;**

**ptr++;**

**}**

**printf(“Array Elements : “);**

**for(i=1; i<=n; i++)**

**{**

**Printf(“%d”,\*p);**

**p++;**

**}**

**printf(“addition is %d”, sum);**

**}**

**calloc()**

**malloc ()creates only one block.. while calloc() can create multiple blocks.**

**calloc() can be used for arrays.**

**void \*calloc( number\_of\_blocks, size for each block in bytes);**

**Syntax :**

**pointer = (Data\_Type\*) calloc(n,Size in bytes);**

**// here function calloc() returns the address of first byte of first block.**

**// malloc() has garbage value in all variables while calloc initionalize with 0.**

**// returns null pointer if block is not created successfully.**

**Example Programm for calloc() in C**

**#include<stdio.h>**

**#include<stdlib.h> // malloc(), calloc() and other functions are here in this file.**

**main()**

**{**

**int n, \*ptr, \*p, i, sum=0;**

**printf( “number of elements to be entered”);**

**scanf(“%d”,&n);**

**ptr = (int \*)calloc(n, sizeof(int));**

**p= ptr;**

**if(ptr == NULL)**

**{**

**printf(“memory block is not created successfully);**

**exit(0); // 0 means normal termination.**

**}**

**printf(“enter %d elements”,n);**

**for(i =1; i<=n; i++)**

**{**

**scanf(“%d”,ptr);**

**sum = sum + \*ptr;**

**ptr++;**

**}**

**printf(“Elements are “);**

**for(i =1; i<=n; i++)**

**{**

**printf( “%d”, \*p);**

**p++;**

**}**

**printf(“ Addition = %d”,sum);**

**free(ptr); // free can be used to free the memory so that we can use that memory in other program.**

**}**

**// calloc has 2 arguments and can create more than 1 block.**

**}**

**Realloc()**

**realloc() function is used to change the size of the memory which is allocated by malloc() or alloc().**

**You can increase / decrease the size of memory using realloc().**

**It returns void pointer.**

**Syntax : void \* realloc(void \*ptr, NewSizeInBytes);**

**Here ptr: is old pointer by which the memory allocation is done using malloc or calloc function.**

**Pointer = (cast\_type\*) realloc( ptr, New\_Size\_in\_bytes);**

**=====\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*=================\*\*\*\*\*\*\*\*\*\*\*\*\***

* Write a program that stores names of the best hospitals in Riyadh into an array of strings.
	+ Define a constant variable MAX and make it equal to 5.
	+ Use the main code provided in the end of this question. Which shows a menu where the user will choose one of the 4 options:
		- Add a new hospital name: Which calls AddName function.
		- Delete a hospital name: Which calls RemoveName function.
		- Print the hospitals names: Which calls PrintNames function.
		- Exit: Which terminates the program.
* Write the following functions:
	+ Write the function ***AddName*** that takes an array of strings pointers called Names and a pointer of integer size.
		- Check if there is still enough space to store a new name.
		- Hint: you will need to use the value of MAX to check.
		- If there is a space, ask the user to input the hospital name and store it in a huge array of char (70 char).
		- Calculate the length of the hospital name.
		- Allocate a dynamic memory to store the entered hospital name and store its location in one of Names indexes.
		- Increment the size by one.
			* void AddName(char \*Names[],int \*size)
	+ Write the function ***RemoveName*** that takes an array of strings pointers called Names and a pointer of integer size.
		- Check if the array is not empty.
		- If it’s not, asks the user to input the index of the hospital name that he wants to delete. Assume that the user will enter indices starting from 0.
		- If the entered index is within a correct range of indices, Free the dynamically allocated memory.
		- Shift left all the hospitals names that comes after it.
		- Decrement the size by one.
			* void RemoveName(char \*Names[],int \*size)
	+ Write the function ***PrintNames*** that takes an array of strings pointers called Names and an integer size. Then prints all of the names separated by commas (,).
		- Hint: Make sure that the array is not empty before printing.
		- void PrintNames(char \*Names[],int size)

**Model Answer:**

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 5
void AddName(char \*[],int \*);
void RemoveName(char \*[],int \*);
void PrintNames(char \*[],int);

int main()

{

 char \*Names[MAX];

 int size = 0;

 int c;

 do{

 printf("=========================\n");

 printf("1- Add a new name.\n");

 printf("2- Delete an old name.\n");

 printf("3- Print names.\n");

 printf("4- Exit.\n");

 printf("=========================\n");

 printf("Enter your choice: ");

 scanf("%i", &c);

 printf("=========================\n");

 switch(c){

 case 1:
 AddName(Names,&size);

 break;

 case 2:
 RemoveName(Names,&size);

 break;

 case 3:
 PrintNames(Names,size);

 break;

 case 4:
 printf("Good bye.\n");

 break;

 default:
 printf("ERROR: Bad input.\n");

 }

 }while(c != 4);

 }

void AddName(char \*Names[],int \*size)
{
 int Copysize = \*size;
 char \*s;
 if (Copysize >= MAX)
 printf("\n ERROR: Array is full. Cannot add.");
 else
 {
 int i,length=0;
 char name[100];
 printf("Enter the name: ");
 scanf("%s",name);
 for(i=0; name[i]!='\0'; i++)
 length++;
 s = (char \*)malloc((length+1)\*sizeof(char));
 strcpy(s, name);
 Names[\*size]=s;
 \*size=\*size+1;
 printf("\n The entered data has been added successfully.\n");
 }

 }

void RemoveName(char \*Names[],int \*size)
{
if (\*size == 0)
printf("There are no data to delete");
else
{
int index, i;
printf("Please Enter the index of the element you want to delete starting from 0 ");
scanf("%d", &index);
if(index<0 || index >= MAX){
printf("The entered index is incorrect");
return;
}
if(index >= \*size){
printf("The entered index is already free, There is nothing to be deleted");
return;
}
free(Names[index]);
for (i=index; i<MAX-1; i++)
{
Names[i] = Names[i+1];
}[MAX-1] = NULL;
\*size= \*size-1;
printf("Deletion is done successfully");
}
}

void PrintNames(char \*Names[],int size)
{
int i;
if (size>0){
for(i=0; i<size; i++)
printf("%s , ",\*(Names+i));("\n");
}
else
{
printf("There are No data to print");
printf("\n");
}
}