```cpp
//Add two numpers using function-pointer concept

#include<iostream>
using namespace std;

float add(int , float);
int main(){
 float (*fp)(int,float); //function pointer declaration

 //At one time fp stores address of only one function .....here stores address of add function
 fp =&add;  //initialization


 int x =5;
 float y= 6.5;
 float result = (*fp)(x,y);   //without pointer ..... result = add(5,6.5);
 cout<<"Result = "<<result;
 return 0;

}

float add(int a, float b){
 return a+b;
}
```

```cpp
//Swap two numbers using function pointer concept
//Arguments passed is not a pointer

#include<iostream>
using namespace std;

float swap(float , float);
int main(){
 //function pointer declaration
 float (*fp)(float,float);

 //At one time fp stores address
 //of only one function .....here stores
 //address of add function
 fp =&swap; //initialization

 float x =5.9;
 float y= 6.5;
 cout<<"Before Swap :"<<x<<" "<<y<<endl;
 (*fp)(x,y); //without pointer ..... swap(x,y);

 return 0;
}

float swap(float a, float b){
 float temp;
 temp =a;
 a=b;
 b=temp;
 cout<<"After Swap :"<<a<<" "<<b<<endl;
 return 0;
}
```

```cpp
//Swap two numbers using function pointer concept
//Argument passed is a pointer

#include<iostream>
using namespace std;

float swap(float *, float *);
int main(){
 //function pointer declaration
 float (*fp)(float *,float *);

 //At one time fp stores address
 //of only one function .....here stores
 //address of add function
 fp =&swap; //initialization

 float x =5;
 float y= 6.5;
 cout<<"Before Swap :"<<x<<" "<<y<<endl;
 (*fp)(&x,&y); //without pointer ..... swap(x,y);

 return 0;
}

float swap(float *a, float *b){
 float temp;
 temp = *a;
 *a = *b;
 *b = temp;
 cout<<"After Swap :"<<*a<<" "<<*b<<endl;
 return 0;
}
```

```cpp
/*P9.38 Array of function pointers*/
#include<iostream>
using namespace std;

float add(float,int);
float sub(float,int);
float mul(float,int);
float div(float,int);

int main()
{
 int i,b;
 float a;
 float(*fp[])(float,int)={add,sub,mul,div};      //Array of function pointers
 char *operation[]={"Add","Subtract","Multiply","Divide"};   //Pointer to array of character

 cout<<"Enter a float and a integer : ";
  cin>>a>>b;

 for(i=0;i<4;i++)
  cout<<operation[i]<<": "<<(*fp[i])(a,b)<<endl;
 return 0;
}
float add(float a,int b){
 return a+b;
}

float sub(float a,int b){
 return a-b;
}

float mul(float a,int b){
 return a*b;
}

float div(float a,int b){
 return a/b;
}
```

```cpp
//dynamic memory allocation
//for single int ,float,double,char

/*new.... delete
1. data-type pointer-variable = new data-type(value);
2. delete variable-name;

*/

#include<iostream>
using namespace std;

int main()
{

 /* What if enough memory is not available during runtime?
 If enough memory is not available in the heap to allocate,
 the new request indicates failure by throwing an exception
 of type std::bad_alloc, unless nothrow is used with the
 new operator, in which case it returns a NULL pointer */
 int *p = new(nothrow) int(25);
 if (!p){
     cout << "Memory allocation failed\n";
 }

 float *q = new float(75.25);
 char *c = new char('b');
 cout<<"Values at p , q and r are :"<<endl;
 cout<<*p<<" "<<*q<<" "<<*c;

 //erase everything from heap after new operation
 delete p;
 delete q;
 delete c;

 return 0;
}
```

```cpp
//Dynamic memory allocation
//for array  of integers ,float ..

/*new.... delete
1. data_type pointer-variable =new data-type[size-of-aray]
2. delete [] pointer-variable

*/

#include<iostream>
using namespace std;

int main()
{
 //for integer ------------------------------------------------>
 cout<<"FOR INTEGERS"<<endl;
 cout<<"How many elemets you want to enter ?";
 int n;
 cin>>n;
 cout<<endl;

 int *arr =new int[n];
 cout<<"Enter "<<n<<" values for an array"<<endl;
 for(int i=0; i<n; i++){
   cin>>arr[i];
 }

 cout<<"Entered value for an array is :"<<endl;
 for(int i=0; i<n; i++){
   cout<<arr[i]<<" ";
 }
 cout<<endl<<endl;
 delete [] arr;

 return 0;
}
```

```cpp
//Dynamic memory allocation
//for array  of  characters

/*new.... delete
1. data_type pointer-variable =new data-type[size-of-aray]
2. delete [] pointer-variable

*/

#include<iostream>
#include<string>
using namespace std;

int main()
{
 //for character ----------------------------------------------->
 cout<<"FOR CHARACTERS"<<endl;
  cout<<"How many character you want to enter ?";
 int p;
 cin>>p;
 cout<<endl;

 char *cap =new char[p];
 cout<<"Enter "<<p<<" characters for an array"<<endl;
 for(int i=0; i<p; i++){
   cin>>cap[i];
 }

 cout<<"Entered  characters are  :"<<endl;
 for(int i=0; i<p; i++){
   cout<<cap[i]<<" ";
 }
 cout<<endl<<endl;
 delete [] cap;

 return 0;

}
```

```cpp
//Dynamic memory allocation
//for array  of  strings ..

/*new.... delete
1. data_type pointer-variable =new data-type[size-of-aray]
2. delete [] pointer-variable

*/

#include<iostream>
#include<string>
using namespace std;

int main()
{
 //for string
 cout<<"FOR STRINGS"<<endl;
  cout<<"How many string you want to enter ?";
  int q;
  cin>>q;
 cout<<endl;

 string *str =new string[q];
 cout<<"Enter "<<q<<" strings for an array"<<endl;
 for(int i=0; i<q; i++){
  getline(cin,str[i]);
 }

 cout<<"Entered strings  are  :"<<endl;
 for(int i=0; i<q; i++){
   cout<<str[i]<<" ";
 }

 cout<<endl<<endl;
 delete [] str;

 return 0;
}
```

```cpp
//Dynamic memory allocation
//for array  of   structures ..

/*new.... delete
1. data_type pointer-variable =new data-type[size-of-aray]
2. delete [] pointer-variable

*/

#include<iostream>
#include<string>
using namespace std;

struct student{
 string name;
 int rollno;
 char address[20];
};

int main()
{

 int i,n;
 cout<<"How many students data you want to enter ?";
 cin>>n;

 struct student *stuarr = new student[n];
 //struct student stuarr[n]; //------------------->without DMA
 cout<<"Enter name, rollno, address respectively for "<<3<<" student"<<endl;
 for(i=0; i<n; i++){
  cin>>stuarr[i].name>>stuarr[i].rollno>>stuarr[i].address;
 }
 cout<<endl<<endl;

 cout<<"Entered data are:"<<endl;
 for(i=0; i<n; i++){
     cout<<stuarr[i].name<<" "<<stuarr[i].rollno<<" "<<stuarr[i].address;
     cout<<endl;
 }
 return 0;
}
```

```c
#include <stdio.h>

struct node {
 int data1;
 char data2;
 struct node* link;
};

int main()
{
 struct node ob1; // Node1
 // Initialization
 ob1.link = NULL;
 ob1.data1 = 10;
 ob1.data2 = 20;

 struct node ob2; // Node2
 // Initialization
 ob2.link = NULL;
 ob2.data1 = 30;
 ob2.data2 = 40;

 // Linking ob1 and ob2
 ob1.link = &ob2;
 // Accessing data members of ob2 using ob1
 printf("%d", ob1.link->data1);
 printf("\n%d", ob1.link->data2);
 return 0;
}
```