Sample C Programming Questions

1. Calculate the sum of all the elements of an array using pointers

```c
#include<stdio.h>

int main(void)
{     // manipulate pointers to access all array elements
   int numArray[10];
   int i ;
   int sum = 0;
   int *ptr;

   printf("\nEnter 10 elements : ");

   for (i = 0; i < 10; i++)
      scanf("%d", &numArray[i]);   // get input from  the scanf

   ptr = numArray; /* point to the first array element */

   for (i = 0; i < 10; i++)
   {
      sum = sum + *ptr;
      ptr++; // increment pointer to the next array element
   }

   printf("The sum of array elements : %d", sum);

   return 0;
}
```

Sample C Programming Questions

2. Get the maximum element in the array using pointers

```c
#define MAX_SIZE 10
void main () {
    int  arr[MAX_SIZE] = {10, 100, 200, -1, 1000, 500, 20, 60, -50, 26};
    int i;
    int *ptr[MAX_SIZE]; // array of pointers
    int MAX = 0;

    for ( i = 0; i < MAX_SIZE; i++)
    {
            ptr[i] = &arr[i]; // assign the address of each array element to a pointer element
    }
    for ( i = 0; i < MAX_SIZE; i++)
    {
        // print all the array elements using array of pointers
        printf("Value of arr[%d] = %d\n", i, *ptr[i] );
    }

    MAX = *ptr[0]; // assume that the maximum number is the first array element

    for ( i = 1; i < MAX_SIZE; i++)
    {
        if(MAX < *ptr[i])
        {
            MAX = *ptr[i];  }
    }
    printf("Maximum Number = %d\n", MAX );
}
```

Output: *Maximum Number = 1000*

Sample C Programming Questions

3. Traverse the array and print the addresses & values of all elements.

```c
int main()
{
    int i;
    int arr[4] = {1,2,3,4}; //arr ( array name) is a constant pointer to array (cannot be altered)
    int *p_arr = arr; // save the pointer to array in another pointer so we can alter it
    printf("Array Addresses:\n");
    for(i=0;i<4;i++)
    {
        printf("%p\t",arr+i);
    }
    printf("\nFirst Element Address: %p\n",p_arr);
    printf("\nFirst Element value: %d\n",*p_arr);
    p_arr++;
    printf("Second Element Address: %p\n",p_arr);
    printf("\n second Element value: %d\n",*p_arr);
    p_arr+=2;
    printf("Forth Element Address: %p\n",p_arr);
    printf("\n Fourth Element value: %d\n",*p_arr);
    return 0;
}
```

Output:

First element Address 0x00000000, First element Value = 1

Second element Address 0x00000004, Second element Value = 2

Fourth element Address 0x00000012, Fourth element Value = 4

Sample C Programming Questions

4. Implement a function called string_ln to get the length of an input string using pointers

```
int string_ln(char*); // function prototype

int main(void)
{
    char str[20];
    int length;

    printf("\nEnter your string : ");
    gets(str);  // get the string from the scanf

    length = string_ln(str);
    printf("The length of the given string \"%s\" is : %d", str, length);
    return 0;
}

int string_ln(char *ptr) /* ptr = &str[0] */
{
    int str_count = 0;
    while (*ptr != '\0') // loop until the end of the string
    {
        str_count++;
        ptr++; // increment the pointer to get the next character
    }
    return str_count;
}
```

Sample C Programming Questions

5. Trace the code snippet and figure out the print statement

```c
int main(void)
{
    int x = 10;
    int *ptr = &x;
    *ptr = 15; // x = 15
    printf("*ptr=%d \t x=%d\n",*ptr,x);
    printf("*&ptr=%p",*&ptr);
    return 0;
}
```

Output:

ptr = 15 , x = 15 ,

**&ptr = 0x000000000 // pass by address*

6. Trace The code snippet and figure out the print statement

```c
// Given that  &x is 0x00000004 ,  &ptr_1 is 0x00000008 ,
//         &ptr_2 is 0x00000012 and  &ptr_3 is 0x00000016

int main( void) {
  int x = 2 ;
  int *ptr_1 = &x ;
  int **ptr_2 = &ptr_1 ;
  int * ptr_3 = ptr_1 ;
  printf( "ptr_3 %d", ptr_3) ;
  printf( "*ptr_3 %d", x) ;
  }
```

Output: *ptr_3 = 0x00000004 , *ptr_3 = 2*

Sample C Programming Questions

7. Implement an array of student and making use of string.h library

```c
#include<stdio.h>
#include<string.h> //for strcpy function

struct student
{
    int id;
    char name[30];
}record[2]; // array of structures

int main(void)
{
    // 1st student's record
    record[0].id=1;
    strcpy(record[0].name, "Khaled");
    // 2nd student's record
    record[1].id=2;
    strcpy(record[1].name, "Ayman");
    printf("%d %s\n%d %s", record[0].id, record[0].name,record[1].id, record[1].name);
    return 0;
}
```

8. Swapping two pointers using pointer to pointer

```c
#include <stdio.h>

void Swap_two_pointers(int **p1,int **p2)
{
    int *temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}

int main(void)
{
    int num1=5,num2=7;
    int *ptr1 = &num1;
    int *ptr2 = &num2;

    printf("*prt1 = %d\n",*ptr1); /* It should be the value of num1 */
    printf("*prt2 = %d\n",*ptr2); /* It should be the value of num2 */

    Swap_two_pointers(&ptr1,&ptr2);

    printf("*prt1 = %d\n",*ptr1); /* It should be the value of num2 */
    printf("*prt2 = %d\n",*ptr2); /* It should be the value of num1 */

    return 0;
}
```

Sample C Programming Questions

9. Use pointer to structure to access the struct data

```c
struct Student {
    int roll_no;
    char name[30];
    char branch[40];
    int batch;
};

int main()
{

    struct Student s1;
    struct Student* ptr = &s1;

    s1.roll_no = 27;
    strcpy(s1.name, "Kamlesh Joshi");
    strcpy(s1.branch, "Computer Science And Engineering");
    s1.batch = 2019;

    printf("Roll Number: %d\n", (*ptr).roll_no);
    printf("Name: %s\n", (*ptr).name);
    printf("Branch: %s\n", (*ptr).branch);
    printf("Batch: %d", (*ptr).batch);

    return 0;
}
```

Sample C Programming Questions

10. Pointer to array vs pointer to the first element of array

```c
#include<stdio.h>

int main()
{
    // Pointer to an integer
    int *p; // pointer to integer

    // Pointer to an array of 5 integers
    int (*ptr)[5]; // pointer to array of 5 integers
    int arr[5];

    // Points to 0th element of the arr.
    p = arr;

    // Points to the whole array arr.
    ptr = &arr;

    printf("p = %p, ptr = %p\n", p, ptr);

    p++;
    ptr++;

    printf("p = %p, ptr = %p\n", p, ptr);

    return 0;
}
```

Output: *p = 0x7fff4f32fd50, ptr = 0x7fff4f32fd50*

p = 0x7fff4f32fd54, ptr = 0x7fff4f32fd64

p: is pointer to 0^{th} element of the array arr, while ptr is a pointer that points to the whole array arr.

The base type of p is int while base type of ptr is ‘an array of 5 integers’ so it increments by 5*4 bytes = 20 bytes while the p pointer is a base of integer so it increments by 4 bytes only.

We know that the pointer arithmetic is performed relative to the base size, so if we write ptr++, then the pointer ptr will be shifted forward by 20 bytes.