Assignment - 22
A Job Ready Bootcamp in C++, DSA and IOT MySirG
DMA

1. Define a function to input variable length string and store it in an array without memory wastage.

Program -

```
#include <stdio.h>
#include<stdlib.h>
#include <string.h>

char *inputString(char*,int);

int main()
{
    char str[50], *p;

    printf("Enter a string: ");
    fgets(str,50,stdin);

    str[strlen(str)-1] ='\0';

    p = inputString(str,strlen(str));
    printf("Entered string is: %s",p);

    return 0;
}

char *inputString(char *s, int len)
{
   char *ptr = (char*)calloc(len+1,sizeof(char));
   strcpy(ptr,s);
   return ptr;
}
```

Output -

```
Enter a string: C++ Bootcamp
Entered string is: C++ Bootcamp
```

---

2. Write a program to ask the user to input a number of data values he would like to enter then create an array dynamically to accommodate the data values. Now take the input from the user and display the average of data values.

Program -

```
#include <stdio.h>
#include<stdlib.h>
```

```
int main()
{
    int n, *ptr, i;
    float sum = 0.0, avg;
    printf("Enter number of values you want to store: ");
    scanf("%d",&n);

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

    printf("Enter %d array values -\n",n);
    for(i = 0 ; i < n ; i++)
    {
        scanf("%d",(ptr+i));
        sum += *(ptr+i);
    }
    avg = sum / n;

    printf("Average is: %f",avg);
    return 0;
}
```

Output -

```
Enter number of values you want to store: 5
Enter 5 array values -
96 21 44 23 32
Average is: 43.200001
```

---

3. Write a program to calculate the sum of n numbers entered by the user using malloc and free.

Program -

```
#include <stdio.h>
#include<stdlib.h>

int main()
{
    int n, *ptr, i, sum = 0;
    printf("Enter number of values you want to store: ");
    scanf("%d",&n);

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

    printf("Enter %d array values -\n",n);
    for(i = 0 ; i < n ; i++)
    {
        scanf("%d",(ptr+i));
        sum += *(ptr+i);
    }

    printf("Sum is: %d",sum);
    return 0;
}
```

Output -

```
Enter number of values you want to store: 5
Enter 5 array values -
10 20 35 60 40
Sum is: 165
```

---

4. Write a program to input and print text using dynamic memory allocation.

Program -

```
#include <stdio.h>
#include<stdlib.h>
#include <string.h>

int main()
{
    char *strptr;

    strptr = (char*)calloc(50,sizeof(char));

    printf("Enter a text: ");
    fgets(strptr,50,stdin);

    strptr[strlen(strptr)-1] = '\0';

    printf("Entered text is: %s",strptr);
    return 0;
}
```

Output -

```
Enter a text: A quick brown fox jumps over the lazy dog.
Entered text is: A quick brown fox jumps over the lazy dog.
```

---

5. Write a program to read a one dimensional array, print sum of all elements along with inputted array elements using dynamic memory allocation.

Program -

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int arr[5], i;
    int *sum = (int*)malloc(sizeof(int));

    *sum = 0;
    printf("Enter 5 array values -\n");
    for(i = 0; i < 5; i++)
    {
        scanf("%d",&arr[i]);
        *sum += arr[i];
    }

    printf("Entered array values are -\n");
    for(i = 0; i < 5; i++)
        printf("%d ",arr[i]);

    printf("\nSum is: %d",*sum);
    return 0;
}
```

Output -

```
Enter 5 array values -
1 2 3 4 5
Entered array values are -
1 2 3 4 5
Sum is: 15
```

---

6. Write a program in C to find the largest element using Dynamic Memory Allocation.

Program -

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int *p = (int*)calloc(5,sizeof(int));
    int i;

    printf("Enter 5 array elements -\n");
    for(i = 0; i < 5; i++)
        scanf("%d",p + i);

    int largest = *p;

    for(i = 1; i < 5; i++)
    {
        if(*(p + i) > largest)
            largest = *(p + i);
    }

    printf("Largest element is: %d",largest);
    return 0;
}
```

Output -

```
Enter 5 array elements -
10 -4 34 12 0
Largest element is: 34
```

---

7. Write a program to demonstrate memory leak in C.

Program -

```
#include<stdio.h>
#include<stdlib.h>

void fun(int);

int main()
{
    fun(6);
    return 0;
}

void fun(int n)
{
    int *ptr = (int*)malloc(sizeof(int));
    *ptr = n;
    printf("Value of dynamically created variable is: %d",*ptr);
    // After the complete execution of this function
    // the pointer variable *ptr will get destroyed
    // and the memory created dynamically will be left
    // till the end of the program resulting in memory leak
}
```

---

8. Write a program to demonstrate dangling pointers in C.

Program -

```
#include<stdio.h>
#include<stdlib.h>

int *fun();
int main()
{
    int *p;
    p = fun(); // Dangling pointer
    printf("Value of *p = %d",*p);
    return 0;
}

int *fun()
{
    int *ptr = (int*)malloc(sizeof(int));
    *ptr = 5;
    free(ptr);
    /* Before returning the address of the dynamically created memory
block,
    We are releasing it's memory using free() function and then
returning it's address to a pointer variable which is now pointing to
a memory block which doesn't exist anymore which
    makes that pointer a dangling pointer
    */
    return ptr;
}
```

Output -

```
Value of *p = 0
```

---

9. Write a program to allocate memory dynamically of the size in bytes entered by the user. Also handle the case when memory allocation is failed.

Program -

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int bytes;

    printf("Enter the number of bytes: ");
    scanf("%d",&bytes);

    while(bytes % sizeof(int) != 0)
    {
        bytes += 1;
    }
    int *ptr = (int*)malloc(sizeof(bytes));

    printf("A memory block of %d bytes has been created!",bytes);

    return 0;
}
```

Output -

```
Enter the number of bytes: 6
A memory block of 8 bytes has been created!
```

---

10. Find out the maximum and minimum from an array using dynamic memory allocation in C.

Program -

```
#include<stdio.h>
#include<stdlib.h>
```

```
int main()
{
    int *p = (int*)calloc(5,sizeof(int));
    int i;

    printf("Enter 5 array elements -\n");
    for(i = 0; i < 5; i++)
        scanf("%d",p + i);

    int max = *p, min = *p;

    for(i = 1; i < 5; i++)
    {
        if(*(p + i) > max)
            max = *(p + i);
        if(*(p + i) < min)
            min = *(p + i);
    }
    printf("Maximum element is: %d\nMinimum element is: %d",max,min);
    return 0;
}
```
Output -

```
Enter 5 array elements -
0 -5 67 90 23
Maximum element is: 90
Minimum element is: -5
```