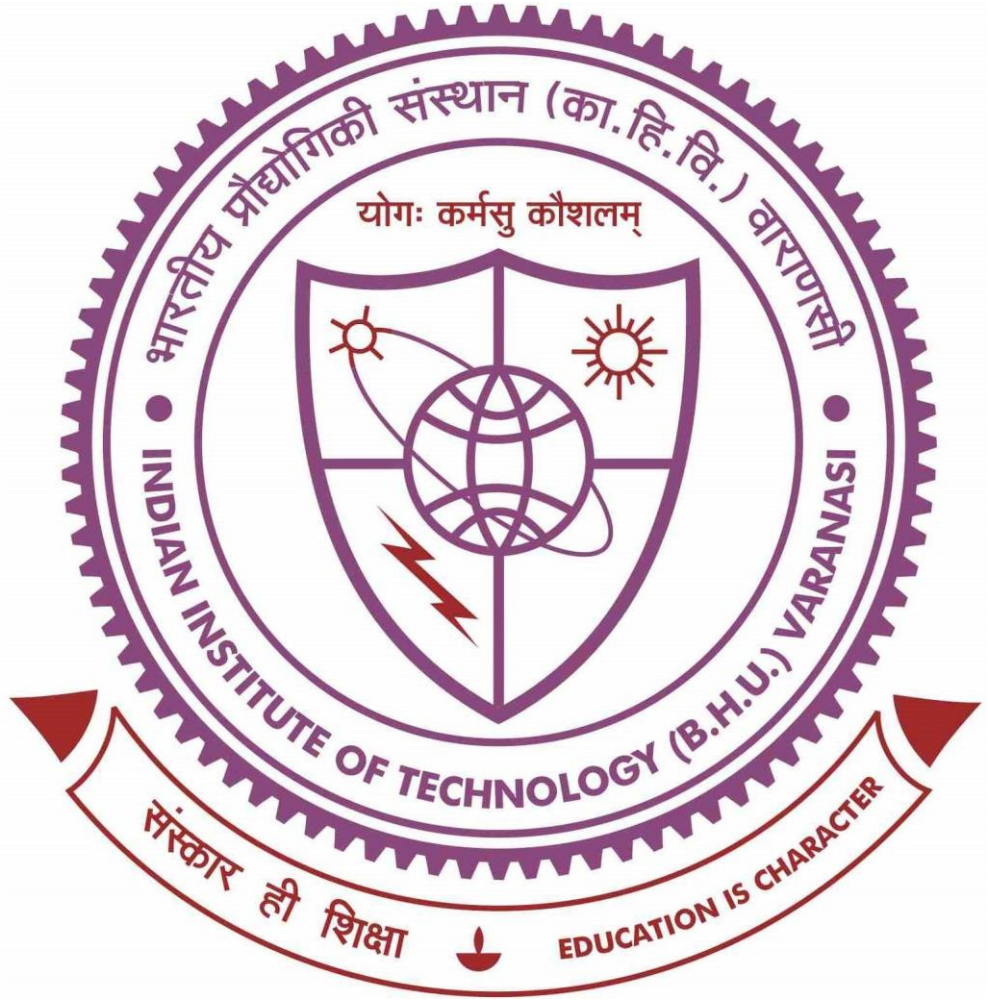

CSO ASSIGNMENT 7

NAME-PARTIK SINGH BUMRAH

SUBJECT-CSO-101

DEPT-CSE

ROLL NO-21075064

EMAIL-

partik.sbumrah.cse21@itbhu.ac.in

1. Use recursive function to find the length of a string.

Sol-

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
int length(char *string);
int main()
{
   char(string[1000];
   printf("Enter the string = ");
   gets(string);
   int len = length(string);
   printf("Length of string = %d", len);
   return 0;
}
int length(char *string)
{
   if (string[0] == '\0')
   {
      return 0;
   }
   else
   {
      char b[1000];
      for (int i = 1;(string[i - 1] != '\0'; i++)
      {
         b[i - 1] = string[i];
      }
      return (1 + length(b));
   }
}
```

2. Compare two strings using recursive functions and print which occursalphabetically first.

Sol-

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
int comparison(char *a, char *b)
{
   strlwr(a);
   strlwr(b);
   if (a[0] < b[0])
   {
      return -1;
   }
   else if (a[0] > b[0])
   {
      return 1;
   }
   else if (a[0] == '\0' && b[0] == '\0')
   {
      return 0;
   }
   else
   {
      char c[1000];
      for (int i = 1; a[i - 1] != '\0'; i++)
      {
         c[i - 1] = a[i];
      }
      char d[1000];
      for (int i = 1; b[i - 1] != '\0'; i++)
      {
         d[i - 1] = b[i];
      }
```

```
        return comparison(c, d);
    }
}
int main()
{
    char a[1000], b[1000];
    printf("Enter the first string : ");
    gets(a);
    fflush(stdin);
    printf("Enter the second string : ");
    gets(b);
    fflush(stdin);
    char c[1000], d[1000];
    strcpy(c, a);
    strcpy(d, b);
    int k = comparison(a, b);
    if (k > 0)
    {
        printf("%s occurs alphabetically first.\n", d);
    }
    else if (k < 0)
    {
        printf("%s occurs alphabetically first.\n", c);
    }
    else if (k == 0)
    {
        printf("Both occur alphabetically together.\n");
    }
    return 0;
}
```

3. Use recursive function to find product of digits of a number.

Sol-

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
int digit_product(int n)
{
   if (n == 0)
   {
      return 1;
   }
   else
   {
      return (n % 10) * digit_product(n / 10);
   }
}
int main()
{
   int n;
   printf("Give number = ");
   scanf("%d", &n);
   int p;
   if (n > 0)
   {
      p = digit_product(n);
   }
   else if (n < 0)
   {
      p = digit_product(-n);
   }
   else if (n == 0)
   {
      p = 0;
   }
   printf("Product of digits = %d", p);
   return 0;
}
```

4. **Using recursive functions find if a number is prime or not.**

Sol-

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
int prime(int a, int b)
{
   if (b == 1)
   {
      return 0;
   }
   else if (b == 0)
   {
      return -1;
   }
   else
   {
      if (a % b == 0)
      {
         return -1;
      }
      else if (a % b != 0)
      {
         return prime(a, b - 1);
      }
   }
}
int main()
{
   int n;
```

```
   printf("Enter the number = ");
   scanf("%d", &n);
   int k = prime(n, n - 1);
   if (k == 0)
   {
      printf("%d is prime\n", n);
   }
   else
   {
      printf("%d is not prime\n", n);
   }

   return 0;
}
```

5. Find the factors of a number using recursive functions.

Sol-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
void factors(int a, int b)
{
   if (b == 1)
   {
      printf("1\n");
   }
   else
   {
      if (a % b == 0)
```

```
        {
            printf("%d\n", b);
        }
        factors(a, b - 1);
    }
}
int main()
{
    int n;
    printf("Enter the number number = ");
    scanf("%d", &n);
    printf("Factors of %d are:\n", n);
    factors(n, n);
    return 0;
}
```

6. Find the gcd of 2 numbers using recursive functions.

Sol-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
int gcd(int m, int n, int p)
{
    if (m % o == 0 && n % p == 0)
    {
        return p;
    }
    else
    {
        return gcd(m, n, p - 1);
    }
}
```

```c
int main()
{
   int a, b, h;
   printf("Enter Numbers:");
   scanf("%d %d", &a, &b);
   if (a >= b)
   {
      h = gcd(a, b, a);
   }
   else if (a < b)
   {
      h = gcd(a, b, b);
   }
   printf("GCD of %d and %d = %d", a, b, h);
   return 0;
}
```

7. Check if a string is a palindrome of not using recursive functions.

Sol-

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <ctype.h>

void palindrome(char *a, int m, int n)
{
   if (a[m] != a[n])
   {
      printf("It is not a Palindrome.");
   }
   else
```

```
    {
        if (m == n || (n - m) == 1)
        {
            printf("It is a Palindrome");
        }
        else
        {
            palindrome(a, m + 1, n - 1);
        }
    }
}
int main()
{
    char a[1000];
    printf("Enter the string : ");
    gets(a);
    int n = strlen(a);
    palindrome(a, 0, n - 1);
    return 0;
}
```