DIGITAL ASSIGNEMENT 2

NAME: GOTAM SAI VARSHITH

REG.NO: 22BCE1605

1) Write a C program to check whether a number is prime, Armstrong, perfect number or not using functions.

Input:

11

Output:

```
11 is prime number
11 is not an Armstrong number
11 is not a perfect number
```

Answer:

```c
#include <stdio.h>
#include <math.h>
int is_prime(int n);
int is_armstrong(int n);
int is_perfect(int n);
int main() {
    int n;
    printf("Enter an integer: ");
    scanf("%d", &n);
```

```
        if (is_prime(n))
         {
              printf("%d is a prime number\n", n);
         } else
         {
              printf("%d is not a prime number\n", n);
         }

           if (is_armstrong(n))
         {
              printf("%d is an Armstrong number\n", n);
         } else
        {
              printf("%d is not an Armstrong number\n", n);
         }

           if (is_perfect(n))
        {
              printf("%d is a perfect number\n", n);
        } else
       {
   printf("%d is not a perfect number\n", n
}
  return 0;
 }

 int is_prime(int n) {
    int i;
    if (n <= 1) {
        return 0;
    }
    for (i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
 }
 int is_armstrong(int n) {
    int sum = 0, temp = n, digits = 0;
    while (temp > 0) {
        digits++;
        temp /= 10;
    }
    temp = n;
    while (temp > 0) {
        int remainder = temp % 10;
        sum += pow(remainder, digits);
        temp /= 10;
    }
    return (sum == n);
}


int is_perfect(int n) {
    int i, sum = 0;
    for (i = 1; i < n; i++) {
        if (n % i == 0) {
            sum += i;
        }
    }
    return (sum == n);
}
```

2) Write a c program to find the number of words, vowels, consonants, space and special characters in a string
   INPUT:

*Nothing is impossible in this world.

OUTPUT:

```
Words = 6
Vowels = 10
Consonants = 20Space = 5
Special Characters = 2
```

Answer:

```
#include <stdio.h>
#include <ctype.h>
int main() {
    char str[1000];
    int words = 0, vowels = 0, consonants = 0, spaces =
0, special_chars = 0;
    int i;
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    for (i = 0; str[i] != '\0'; i++) {
        if (isalpha(str[i])) {
            if (tolower(str[i]) == 'a' || tolower(str[i]) == 'e'
|| tolower(str[i]) == 'i' || tolower(str[i]) == 'o' ||
tolower(str[i]) == 'u') {
                vowels++;
            } else {
                consonants++;
            }
        } else if (isspace(str[i])) {
            words++;
            spaces++;
        } else {
            special_chars++;
```

```
 }
    }
   words++;
    printf("Words = %d\n", words);
    printf("Vowels = %d\n", vowels);
   printf("Consonants = %d\n", consonants);
   printf("Space = %d\n", spaces);
    printf("Special Characters = %d\n", special_chars);
       return 0;
   }
```

3) Write a C program that accepts a string as input, print the length of the string and display the word frequency, then use pointers to find the first repeated and non-repeated character in the string, and print the output:

POSSIBLE TEST CASES:

```
INPUT:
SUJITHRA
OUTPUT:
Length of the string is: 8
Word frequency is: 8
No repeated characters found in the string.
First non-repeated character is: S
#2 INPUT:
ASSDFG
OUTPUT:

Length of the string is: 6

Word frequency is: 5
```

First repeated character is: S

First non-repeated character is: A

#3 INPUT:

RUDRESH

OUTPUT:

```
Length of the string is: 7
Word frequency is: 6
First repeated character is: R
First non-repeated character is: U
```

Answer:

```
#include <stdio.h>
#include <string.h>
#define MAX_LENGTH 100
int main() {
    char str[MAX_LENGTH];
    int len, freq[256] = {0}, i;
    char *p, *rep = NULL, *nonrep = NULL;
printf("Enter a string: ");
fgets(str, MAX_LENGTH, stdin);
 len = strlen(str) - 1;
    for (p = str; *p != '\0'; p++) {
        freq[(int)*p]++;
    }
printf("Length of the string is: %d\n", len);
printf("Word frequency is: ");
    for (i = 0; i < 256; i++) {
        if (freq[i] > 0) {
            printf("%c:%d ", i, freq[i]);
        }
    }
    printf("\n");
    for (p = str; *p != '\0'; p++) {
        if (freq[(int)*p] == 1 && nonrep == NULL) {
            nonrep = p;
        } else if (freq[(int)*p] > 1 && rep == NULL) {
            rep = p;
        }
        if (nonrep != NULL && rep != NULL) {
            break;
        }
    }

    if (rep == NULL) {
        printf("No repeated characters found in the
string.\n");
    } else {
        printf("First repeated character is: %c\n", *rep);
    }

    if (nonrep == NULL) {
        printf("No non-repeated characters found in the
string.\n");
    } else {
        printf("First non-repeated character is:
%c\n",*nonrep);
    }

    return 0;}
```

4) Write a cprogram to get the employee information name,age,position and Date of joining. Print the employee list based on Alphabaetical order. Display the order of the employees based on date of joining.
   SAMPLE INPUT MODEL:
   Enter the number of employees: 3
   Enter details of employee 1:
   Name: Jane
   Age: 34
   Position: HR
   Date of joining (dd/mm/yyyy): 10/2/2000
   Enter details of employee 2:
   Name: Amie
   Age: 23
   Position: Sales
   Date of joining (dd/mm/yyyy): 12/03/2004
   Enter details of employee 3:
   Name: Balu
   Age: 45
   Position: Scurity
   Date of joining (dd/mm/yyyy): 1/1/1998
   SAMPLE OUTPUT MODEL:
   Employee List sorted by name:
   Name: Amie
   Age: 23
   Position: Sales
   Date of Joining: 12/03/2004
   Name: Balu
   Age: 45

```
Position: Security
Date of Joining: 1/1/1998
Name: Jane
Age: 34
Position: HR
Date of Joining: 10/2/2000
Employee List sorted by date of joining:
Name: Balu
Age: 45
Position: Security
Date of Joining: 1/1/1998
Name: Jane
Age: 34
Position: HR
Date of Joining: 10/2/2000
Name: Amie
Age: 23
Position: Sales
Date of Joining: 12/03/2004
```

PUBLIC TEST CASE:

```
3
Jane
34
HR
10/2/2000
Amie
23
Sales
12/03/2004
```

```
Balu
45
Security
1/1/1998
OUTPUT:
Employee List sorted by name:
Amie
23
Sales
12/03/2004
Balu
45
Security
1/1/1998
Jane
34
HR
10/2/2000
Employee List sorted by date of joining:
Balu
Jane
Amie
```

Answer:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_EMPLOYEES 100
#define MAX_NAME_LENGTH 50
#define MAX_POSITION_LENGTH 50
```

```
#define DATE_LENGTH 11
struct Employee {
    char name[MAX_NAME_LENGTH];
    int age;
    char position[MAX_POSITION_LENGTH];
    char date[DATE_LENGTH];
};
int compareByName(const void *a, const void *b) {
    const struct Employee *ea = (const struct Employee
*)a;
    const struct Employee *eb = (const struct Employee
*)b;
    return strcmp(ea->name, eb->name);
}
int compareByDate(const void *a, const void *b) {
    const struct Employee *ea = (const struct Employee
*)a;
    const struct Employee *eb = (const struct Employee
*)b;
    return strcmp(ea->date, eb->date);
}
int main() {
    int n, i;
    struct Employee employees[MAX_EMPLOYEES];
printf("Enter the number of employees: ");
    scanf("%d", &n);
    getchar(); // consume newline character
 for (i = 0; i < n; i++) {
        printf("Enter details of employee %d:\n", i+1);
printf("Name: ");
        fgets(employees[i].name, MAX_NAME_LENGTH,
stdin);
        employees[i].name[strcspn(employees[i].name,
"\n")] = '\0';
        printf("Age: ");
        scanf("%d", &employees[i].age);
        getchar();
        printf("Position: ");
        fgets(employees[i].position,
MAX_POSITION_LENGTH, stdin);
employees[i].position[strcspn(employees[i].position,
"\n")] = '\0';
        printf("Date of joining (dd/mm/yyyy): ");
        fgets(employees[i].date, DATE_LENGTH, stdin);
        employees[i].date[strcspn(employees[i].date,
"\n")] = '\0';
    }
qsort(employees, n, sizeof(struct Employee),
compareByName);
    printf("\nEmployee List sorted by name:\n\n");
    for (i = 0; i < n; i++) {
        printf("Name: %s\n", employees[i].name);
        printf("Age: %d\n", employees[i].age);
        printf("Position: %s\n", employees[i].position);
        printf("Date of Joining: %s\n\n",
employees[i].date);
    }
```

```
 qsort(employees, n, sizeof(struct Employee),
compareByDate);
   printf("\nEmployee List sorted by date of
joining:\n\n");
   for (i = 0; i < n; i++) {
      printf("Name: %s\n", employees[i].name);
   }

   return 0;
}
```

5) Get the three angles of a triangle as input.
find the count of the type of the triangle.
Continue the process for 5 times.
If the sum of the three angles is greater than 180 then prompt for correct values. (the sum of all internal angles of a triangle is always equal to 180°). Keep the count of the wrong entries also.
Acute Angled Triangle (all three angles less than 90°)
Right-Angled Triangle (one angle that measures exactly 90°)
Obtuse Angled Triangle (one angle that measures more than 90°)
Sample i/p:

```
60
70
50
40
50
90
```

40

40

100

30

30

120

90

60

30

Sample o/p:

Acute Angled Triangle: 1

Right Angled Triangle: 2

Obtuse Angled Triangle: 2

Wrong Entries: 0

Second Sample i/p:

60

70

50

40

50

90

40

40

100

30

30

120

90

90

30

Wrong Entry try again

90

30

60

Sample o/p:

```
Wrong Entry try again
Acute Angled Triangle: 1
Right Angled Triangle: 2
Obtuse Angled Triangle: 2
Wrong Entries: 1
```

Answer:

```
#include <stdio.h>
int main() {
int i, j, a, b, c, sum, acute = 0, right = 0, obtuse = 0,
wrong = 0;
  for (i = 1; i <= 5; i++) {
    printf("Enter the three angles of triangle
%d:\n",i);
    scanf("%d %d %d", &a, &b, &c);
    sum = a + b + c;
    if (sum > 180) {
      printf("Wrong Entry try again\n");
      wrong++;
      i--;
      continue;
    }
    if (a < b) {
      j = a;
```

```
        a = b;
            b = j;
        }
        if (a < c) {
            j = a;
            a = c;
            c = j;
        }
        if (a*a == b*b + c*c) {
            printf("Right-Angled Triangle\n");
            right++;
        } else if (a*a < b*b + c*c) {
            printf("Acute Angled Triangle\n");
            acute++;
        } else {
            printf("Obtuse Angled Triangle\n");
            obtuse++;
        }
    }
    printf("\nAcute Angled Triangle: %d\n", acute);
    printf("Right Angled Triangle: %d\n", right);
    printf("Obtuse Angled Triangle: %d\n", obtuse);
    printf("Wrong Entries: %d\n", wrong);
    return 0;
}
```

6) The temperature in Chennai exists somewhere between 280C to 330C in a particular week. The temperature in Delhi is 8 0C lesser than in Chennai. Likewise, the temperature in Haveri is 5 0C more than

that of Chennai. Write a C program to find the temperature of Gangtok for a particular week, which is the temperature difference between Delhi and Haveri. Get the temperature of Chennai (0C) as input for the week of 7 days and the temperature of Gangtok (0C) as output for the week of 7 days. Implement the program using arrays and validate the rules.

Answers:

```
#include <stdio.h>
#define DAYS 7
int main() {
   int chennai[DAYS], delhi[DAYS], haveri[DAYS],
gangtok[DAYS];
   int i;
   printf("Enter the temperature of Chennai for the
week:\n");
   for (i = 0; i < DAYS; i++) {
      scanf("%d", &chennai[i]);
      if (chennai[i] < 28 || chennai[i] > 33) {
         printf("Invalid temperature for Chennai.
Temperature should be between 28C and 33C.\n");
         return 1;
      }
   }
for (i = 0; i < DAYS; i++) {
      delhi[i] = chennai[i] - 8;
      haveri[i] = chennai[i] + 5;
   }
for (i = 0; i < DAYS; i++) {
        gangtok[i] = haveri[i] - delhi[i];
    }
    printf("\nTemperature of Gangtok for the
week:\n");
    for (i = 0; i < DAYS; i++) {
        printf("%dC ", gangtok[i]);
    }
    printf("\n");
    return 0;
}
```

7) Samantha is an avid collector of lucky numbers. She believes that each number has its own unique energy and can bring good luck if used correctly. One day, she came across a new number that had a mysterious aura and she could not resist finding out more about it. She heard that the number may fall within the digit combinations, so she decided to sum up all the four-digit even numbers and then keep adding the digits of the summation until a single digit is found to unlock its secrets. Later, she must check again whether the single digit is odd or even. If odd, then you must say "Odd Found" otherwise you must return "Even found". Write a C program to help her in finding the mysterious number.

Answer:

```
#include <stdio.h>
int main() {
    int i, j, sum, digit, odd_found = 0;
```

```
  sum = 0;
    for (i = 1000; i < 10000; i += 2) {
        sum += i;
    }
while (sum > 9) {
        digit = 0;
        while (sum > 0) {
            digit += sum % 10;
            sum /= 10;
        }
        sum = digit;
    }
    if (sum % 2 == 1) {
        printf("Odd Found\n");
        odd_found = 1;
    } else {
        printf("Even Found\n");
    }
return odd_found;
}
```

8) A digital locker in the bank is protected with a security mechanism. To open the locker a password of 9 characters is required. The input characters should be accepted as 3x3 matrix and two diagonal characters of the matrix are concatenated (refer to the example given below) and compared with the password already stored in a character array for authentication. Write a C program to implement this logic for password verification.

Input to open the device

 a b c

 d e f

 g h i

Concatenation of Diagonal characters:

 aeiceg

Answer:

```
#include <stdio.h>
#include <string.h>
#define PASSWORD "aeiceg"
int main() {
    char matrix[3][3], diagonal[5];
    int i, j, k = 0;
    printf("Enter the 3x3 matrix:\n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            scanf(" %c", &matrix[i][j]);
        }
    }
    diagonal[k++] = matrix[0][0];
    diagonal[k++] = matrix[1][1];
    diagonal[k++] = matrix[2][2];
    diagonal[k++] = matrix[0][2];
    diagonal[k++] = matrix[2][0];
    if (strcmp(diagonal, PASSWORD) == 0) {
        printf("Password verified\n");
    } else {
        printf("Incorrect password\n");
    }
return 0;
}
```

9) An international school of class 7 with a class strength of 25, decided to assign additional marks for their students in Maths subject, to increase their class average. The additional marks were given to each student based on their month of birth. That is she was to give that number (month of birth) as the booster marks. The teacher wants to find the class average for the original marks as well as for the revised marks. She 10 marks wants to decide whether to implement this revision in marks or not based on the significant improvement in the class average. Write a C program to help the teacher get the class average for the original marks as well as the revised marks. She wants to know whether to implement this revision or not. This is decided based on the condition that the revision should bring a significant increase in the class average of 5 marks. Else, she is not planning to implement this revision in the marks strategy. Write the program to display this decision of "Can implement – Significant increase in class average" or "Need not implement – No significant increase in class average". Keep every operation in this program separate. Get the students' original marks and the month of their birth as input

Answer:

```
#include <stdio.h>
#define CLASS_SIZE 25
```

```
int main() {
   int marks[CLASS_SIZE], birth_month[CLASS_SIZE];
   int i, sum_original = 0, sum_revised = 0;
   float avg_original, avg_revised;
   printf("Enter the original marks and birth months of
the students:\n");
   for (i = 0; i < CLASS_SIZE; i++) {
      scanf("%d %d", &marks[i], &birth_month[i]);
   }
   for (i = 0; i < CLASS_SIZE; i++) {
      sum_original += marks[i];
   }
   avg_original = (float)sum_original / CLASS_SIZE;
   for (i = 0; i < CLASS_SIZE; i++) {
      marks[i] += birth_month[i];
      sum_revised += marks[i];
   }
   avg_revised = (float)sum_revised / CLASS_SIZE;
   if (avg_revised - avg_original >= 5) {
      printf("Can implement - Significant increase in
class average\n");
   } else {
      printf("Need not implement - No significant
increase in class average\n");
   }
return 0;
}
```

10) Paul is provided with a number “x” whose scope will remain throughout the program. Using the

concept of recursion, help Paul to write a “C program” for finding the value of (x)n where n should be less than or equal to 5

Answer:

```
#include <stdio.h>
int power(int x, int n);
int main() {
    int x, n, result;
    printf("Enter x and n (<= 5):\n");
    scanf("%d %d", &x, &n);
    result = power(x, n);
    printf("%d^%d = %d\n", x, n, result);
    return 0;
}
int power(int x, int n) {
    if (n == 0) {
        return 1;
    } else if (n == 1) {
        return x;
    } else if (n <= 5) {
        return x * power(x, n-1);
    } else {
        return -1;
    }
}
```

11) Write a C program to divide the integer array into two halves using function recursion. Call the user-defined “divide” function recursively, with the left half of the split array being passed as an argument for that

function. Let the recursive function to get executed until the array size becomes one. Count the number of iterations to reach the base condition. Explain the working procedure of recursive function with stack structure.

Answer:

```
#include <stdio.h>
void divide(int arr[], int start, int end, int *count);
int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int n = sizeof(arr) / sizeof(arr[0]);
    int count = 0;
    divide(arr, 0, n-1, &count);
    printf("Number of iterations: %d\n", count);
    return 0;
}
void divide(int arr[], int start, int end, int *count) {
    if (start == end) {
        return;
    }
    int mid = (start + end) / 2;
    divide(arr, start, mid, count);
    (*count)++;
    divide(arr, mid+1, end, count);
}
```

12) Consider that you are going to analyze the characters in the given string. Write a C program to extract the characters in the given string and print whether the character is an uppercase alphabet,

lowercase alphabet, digits, whitespace, special symbols. Print the count of each category by storing their counts in an array. Use appropriate looping constructs to implement this.

Answer:

```
#include <stdio.h>
#include <ctype.h>
#define MAX_LENGTH 100
int main() {
    char str[MAX_LENGTH];
    int counts[5] = {0};
    int i;
    printf("Enter a string:\n");
    fgets(str, MAX_LENGTH, stdin);
    for (i = 0; str[i] != '\0'; i++) {
        if (isupper(str[i])) {
            counts[0]++;
        } else if (islower(str[i])) {
            counts[1]++;
        } else if (isdigit(str[i])) {
            counts[2]++;
        } else if (isspace(str[i])) {
            counts[3]++;
        } else {
            counts[4]++;
        }
    }
    printf("Uppercase alphabets: %d\n", counts[0]);
    printf("Lowercase alphabets: %d\n", counts[1]);
    printf("Digits: %d\n", counts[2]);
    printf("Whitespaces: %d\n", counts[3]);
    printf("Special symbols: %d\n", counts[4]);
    return 0;
}
```

13) Write a c program to find the sum of the series 1!/1+2!/2+3!/3+4!/4+5!/5 ... n!/n by utilizing user defined recursive function? Get the value of n from the user. Do not use any storage classes. Without returning the calculated result from the function, display the result in main

Answer:

```
#include <stdio.h>
void sum_series(int n, float *sum);
int main() {
    int n;
    float sum = 0;
    printf("Enter n:\n");
    scanf("%d", &n);
    sum_series(n, &sum);
    printf("Sum of the series: %f\n", sum);
    return 0;
}
void sum_series(int n, float *sum) {
    if (n == 0) {
        return;
    }
    float term = 1;
    int i;
for (i = 1; i <= n; i++) {
        term *= i;
    }
    *sum += term / n;
    sum_series(n-1, sum);
}
```

14) ABC car showroom sells various types of cars such as Hatchback, Sedan, SUVs, and MUV. Due to the year-end sale, the showroom provides a 3%, 5%, 10%, and 15% discount for various car models Hatchback, Sedan, SUV, and MUV respectively. Also applies 12% of GST for the total amount of purchase Write a C program to implement the above scenario which will read the type_of_the_car, price_of_the_car and extra-fitting_price_of_the_car as input from the user and estimate the Net amount to be paid to the showroom. If the type of car is other than Hatchback, Sedan, SUV, and MUV then display “Invalid Type”. (Difficulty Level: Easy)

The net amount to be paid to the showroom is estimated as follows:

(For example-if the purchased car is Hatchback)

Total = price_of_the_car + extra_fitting_price_of_the_car

Discount = Total * 0.03 // 0.03 denotes 3% wastage

gst = (Total - Discount) * 0.12 // 0.12 denotes 12% GST

net = Total – Discount + gst.

Answer:

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

```
#include <string.h>
#define MAX_LENGTH 20
int main() {
    char type[MAX_LENGTH];
    float price, fitting, total, discount, gst, net;
    printf("Enter the type of car (Hatchback, Sedan,
SUV, MUV):\n");
    scanf("%s", type);
    printf("Enter the price of the car:\n");
    scanf("%f", &price);
    printf("Enter the extra fitting price of the car:\n");
    scanf("%f", &fitting);
    total = price + fitting;
    if (strcmp(type, "Hatchback") == 0) {
        discount = total * 0.03;
    } else if (strcmp(type, "Sedan") == 0) {
        discount = total * 0.05;
    } else if (strcmp(type, "SUV") == 0) {
        discount = total * 0.10;
    } else if (strcmp(type, "MUV") == 0) {
        discount = total * 0.15;
    } else {
        printf("Invalid Type\n");
        return 0;
    }
    gst = (total - discount) * 0.12; // 12% GST
    net = total - discount + gst;
    printf("Net amount to be paid: %.2f\n", net);
    return 0;}
```

15) Write a C Program that reads the input as a string from the user in main ().

The input should be a sentence with two words. Pass this string to a function.

Inside the function do the following operations:

- For the first word, keep only the first character of the word to be in upper case and the rest of the characters in lower case.
- For the second word, convert all the characters into upper case letter.
- Print the revised string consisting of the two words in the function itself
- Find the length of the entire string. Print its length in the function itself in the next line of the revised string. Use appropriate string function to print this result in the next line.
- Return the length of the string, if the length is less than 20. Else return the size of the string.

Consider the input string given by the user is:

- "Computer programming"
- Revised string to be printed in the function is "Computer PROGRAMMING"

What is the significant different between length and size of the string? What is the value that will get returned for the input string that is considered?

Answer:

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

```
int process_string(char *str);
int main() {
    char str[100];
    printf("Enter a sentence with two words:\n");
    fgets(str, 100, stdin);
    str[strcspn(str, "\n")] = '\0';
    int len = process_string(str);
    printf("Length/Size of the string: %d\n", len);
    return 0;
}
int process_string(char *str) {
 int len = strlen(str);
    int i;
    for (i = 0; i < len; i++) {
        if (str[i] == ' ') {
            break;
        }
    }
    str[0] = toupper(str[0]);
    for (int j = 1; j < i; j++) {
        str[j] = tolower(str[j]);
    }
    for (int j = i+1; j < len; j++) {
        str[j] = toupper(str[j]);
    }
    printf("Revised string: %s\n", str);
    printf("Length of the string: %ld\n", strlen(str));
    if (len < 20) {
        return strlen(str);
```

```
  } else {
        return len;
    }
}
```