Mandatory Section A

Q1.1. **Mark 2**

Write a function that takes a string as input and prints it.

void print_string (char[] str) { printf (“%s”, str); }

Q1.2. **Mark 2**

Illustrate using a minimal example the concept of recursion.

void f (int n) { return (n==1) ? 1 : n+f(n-1); }

1. here the base case is when n=1
2. the input n to the function should be some positive integer

Q1.3. **Mark 2**

int arr[2][2][2];

printf(“%d %d %d %d”, sizeof (arr), sizeof (arr[1]), sizeof (arr[1][0]), sizeof (arr[1][0][1]));

Write down the output of the printf statement above.

32 16 8 4

Q1.4. **Mark 2**

struct new_type { int a; float b[3]; char name[10];}**;**

struct new_type n1;

printf (“%d %d”, sizeof(n1), sizeof(**n1.**name));

Write down the output of the printf statement above.

28 10

note: the answer 26 10 is acceptable; but it will be 28 in reality, due to padding (?)

Q1.5. **Mark 2**

Create your own structure for storing points in a 4-dimensional space.

struct fourD {double dim[4];};

note: other definitions are also acceptable

Q1.6. **Mark 2**

int fun(int* arr) {   printf (“in fun: %d\n”, sizeof (arr));   }

int main()

{ int arr[10];   printf (“in main: %d\n”, sizeof (arr)); fun (arr); }

Write the output of code above.

in main: 40

in fun: 8

Q1.7. **Mark 2**

Give code to return the absolute value of an integer. e.g. both -5 and 5 become 5.

int abs (int x) { if (x>=0) return x else return -x; }

Q1.8. **Mark 2**

double arr[4];

printf(“%p”, arr); ⇒ gives the output 0x1024

Calculate and write down the address of all the elements of the array arr.

arr[0] →0x1024; arr[1] → 0x1028; arr[2] → 0x102C; arr[1] → 0x1030;

Q1.9. **Mark 2**

Write a preprocessor(e.g. #define, #if, etc.) directive for getting the average of two values.

```
#define(X, Y) (((X)+(Y))/2)
```

Q1.10. **Mark 2**

Write a simple code for opening and closing a file named “abc.txt” in write mode. *Just write the variable declaration(s) and the function call(s), no need to write #include, main, etc.*

```
FILE* fp;
fp = fopen (“abc.txt”, “w”);
fclose (fp);
```

Q1.11.

Mark 2

Given a 2D coordinate position of a point, determine which quadrant the point is in. e.g. (-2,-2) is in the third quadrant, (2,3) is in the first quadrant.

```
void quadrant (int x, int y)
{if (x>0}
```

Q1.12. **Mark 2**

Given an alphabet as input, check whether it is a vowel or a consonant.

```
//Assuming the input alphabet is stored in a character variable named ch
if (    ch == ‘a’ || ch == ‘e’ || ch == ‘i’ || ch == ‘o’ || ch == ‘u’
      || ch == ‘A’ || ch == ‘E’ || ch == ‘I’ || ch == ‘O’ || ch == ‘U’ )
            printf (“vowel”);
else
            printf (“consonant”);
```

Q2.1. **Mark 3**

```
void fun(int x, int y)
{
        if (x>y)
                fun(y+1, x-1);
        else if (x<y)
                fun(y-1, x+1);

        printf("%d %d\n", x, y);
}
```

Write the output when *fun(10,2)* is called.

```
6 6
5 7
8 4
3 9
10 2
```

Explanation:

```
evaluating fun(10,2)
calling fun(3,9)
        evaluating fun(3,9)
        calling fun(8,4)
                evaluating fun(8,4)
                calling fun(5,7)
                        evaluating fun(5,7)
                        calling fun(6,6)
                                evaluating fun(6,6)
                                6 6
                        5 7
                8 4
        3 9
10 2
```

Q2.2. **Mark 3**

Write a program/function to find the number of trailing zeros in a given factorial.

```
int count = 0, x, i; //assuming we are finding the trailing zero for n!
     for (i=1; i<=n; i++)
     {
          for(x=i; x%5==0; x/=5)
               count++;
     }
 printf ("%d", count);
```

Q2.3. **Mark 3**

```
float calc_avg (/*(A)formal parameter(s) for passing array*/);
float calc_avg (int *arr, int size);
int main ()
{
        float arr[] = {10.2, 10, 12, 11, 2, 3, 4, 9, 1, 2, 11.4, 2, 5, 19};
        int size = /*(B)write code for calculating number of elements*/;
        int size = sizeof(arr) / sizeof(arr[0]);
        float avg = calc_avg(/*(C)pass the variable(s)*/);
        float avg = calc_avg(arr, size);
}
```

Complete the above prototype and the corresponding function call for passing the array to the function. Just complete the A, B and C marked above. **Note**: You don't have to define the function, just assume it is already done.

Q2.4. **Mark 3**

Write a program/function to find the sum of the series 1!/(N-1) + 2!/(N-2) + 3!/(N-3) + 4!/(N-4) + …upto N-1 terms. Take N as input form user.

```
int fact (int n)
{
        return (n==1 || n==0) ? 1 : n* fact(n-1);
}
double calc_series (int N)
{
        int i;
        double sum=0;
        for(i=1; i<N; i++)
                sum += (double)fact(i)/(N-i);
        return sum;
}
```

Q2.5. **Mark 3**

You have two arrays of integers, each of size 5. Write code to create another array of size 10 containing all the elements in an alternating fashion.

Example:
Array 1: 10, 12, 14, 16, 18
Array 2: 9, 11, 13, 15, 17
Resultant array: 10, 9, 12, 11, 14, 13, 16, 18, 17

```
//assuming arr1[5] and arr2[5] is already scanned
int arr_result[10];
for (i=0; i<10; i++)
{
        if (i%2 == 0)
                arr_result[i] = arr1[i/2];
        else
                arr_result[i] = arr2[i/2];
}
```

Q2.6. **Mark 3**

Show how to allocate memory using Dynamic memory allocation by allocating memory for an integer array of size N. N is read as input from the user.

```
int N;
int *arr;
scanf (“%d”, &N);
arr = (int*) malloc(sizeof(int) * N);
```

Q3.1. **Mark 5**

Write a program/function that takes a month number (1-12) for the year 2023 as input and prints the dates for the Saturdays and Sundays in the month.

Example:

input 2 ⇒ Output: 4 5 11 12 18 19 25 26

input 9 ⇒ Output: 2 3 9 10 16 17 23 24 30

```
//Jan 1, 2023 was a Sunday
void sat_sun (int m)
{
    int days = 0, d, i;
    int sat = 7, sun =1; //first sat, sun dates of 2023

     //count the number of days before the given month
     for (i=1; i<=m; i++)
     {
          d = days;
          if (i==1 || i==3 || i==5 || i==7 || i==8 || i==10 || i==12)
                days +=  31;
          else if (i==2)
                days += 28;
          else
                days += 30;
     }

     while (sat<d) sat += 7;      //push sat to the first saturday of of target month
     while (sun<d) sun += 7;     // same as above, for sun

//just printing the sundays and then saturdays for the target month
     while (sun<= days)
       {
            printf ("%d ", sun-d);
            sun+=7;
       }
     while (sat<= days)
       {
            printf ("%d ", sat-d);
            sat+=7;
       }
//figure out how to order (i.e. sat sun sat sun, etc.) them yourselves; hint.you can use an array
}
```

Q3.2. **Mark 5**

Write a program/function that takes a string as input and prints the upper case version of the string. Do not use library functions.

```
char str[100]; //assuming input string is less than size 100
int i=0;
scanf (“%s”, str);
while (str[i]!=’\0’)
{
        if (str[i] <=’z’ && str[i] >= ‘a’)
                str[i] = str[i] - (‘a’ – ‘A’);
        i++;
}
printf (“%s”, str);
```

Q3.3. **Mark 5**

Write a C program to divide two integers (dividend and divisor) **without using** *multiplication*(*), *division*(/) and *modulo division*(%) operator.

```
int divident, divisor, result=0, x, y;
scanf(“%d %d”, &dividend, &divisor); //x is the divident and y is the divisor

x = dividend; y = divisor;
while ((x – y)>0)
{
        x = x – y;
        result++;
}
printf (“%d/%d = %d”, dividend,  divisor, result);
```

Q3.4. **Mark 5**

```
typedef struct complex
{
	float real;
	float imaginary
}Q;
```

Write a function that takes the two complex numbers (you can use the above structure) and prints the multiplied value in x + yi format (check examples below, ignore 0s and treat 1i as i).

Hint: (x+yi)*(a+bi) = (ax-by) + (ay+bx)i

```
void print (Q n)
{
	if (n.real != 0)
		printf (“%f”, n.real);

	if (n.imaginary != 0)
	{
		printf (“ %c ”, (n.imaginary < 0) ? ’-’ : ’+’);
		if (n.imaginary != 1 && n.imaginary != -1)
			printf (“%f”, (n.imaginary < 0) ? -n.imaginary : n.imaginary);
		printf (“i”);
	}
}

void multiply(Q c1, Q c2)
{
	Q result;
	result.real = (c1.real * c2.real) – (c1.imaginary * c2.imaginary);
	result.imaginary = (c1.real * c2.imaginary) + (c1.imaginary * c2.real);
	print (result);
}
```

Optional Section B

Q4.1. **Mark 1**

[Python] Give an example of how to print a variable in python.
print(v) #prints the variable v

Q4.2. **Mark 1**

[Python] Give an example of how to do integer division in python. (e.g. 5÷2 = 2)
x//y

Q4.3. **Mark 1**

[Python] Give an example of how to write a list of integers in python.
lst=[3,45,65,6,4,6,7,30]

Q4.4. **Mark 1**

[Python] Give an example of how to assign a value in a dictionary.
d={}
d['key'] = "value"

Q4.5. **Mark 1**

[Python] Give an example of how to access list elements using negative indexes.
lst=[3,45,65,6,4,6,7,30]
lst[-1] will evaluate to 30 (i.e. the last value in the list), lst[-2] will be 7, etc.

Q4.6. **Mark 1**

[Python] A = [10, 12, 14, 16, 18, 20, 22, 24, 26, 28] ⇒ what is A[1:8]?
[12, 14, 16, 18, 20, 22]

Q4.7. **Mark 1**

[Python] A = [10, 12, 14, 16, 18, 20, 22, 24, 26, 28] ⇒ what is A[1:7:2]?
[12, 16, 20]

Q4.8. **Mark 1**

[Python] A = [10, 12, 14, 16, 18, 20, 22, 24, 26, 28] ⇒ how do you reverse it?
A[-1:0:-1]

Q4.9. **Mark 1**

[Python] Assume x = "Hi" and y=”There" ⇒ what will be x+y?
"HiThere"

Q4.10. **Mark 1**

[Python] How do you calculate the length of the string "sly fox"?
len("sly fox")