Indian Association for the Cultivation of Science

(Deemed to be University under *de novo* Category)

Master's/Integrated Master's-PhD Program/Integrated Bachelor's-Master's Program/PhD Course

Mid-Semester Examination-Autumn 2022

Subject: Introduction to Computing with Python ***Subject Code(s): MCS1101B***

Full Marks: 25 ***Time Allotted: 2 h***

Instructions *(please read carefully each point)*

- ★ Write as little as possible without missing out on any details
  - Think carefully before answering
  - There is no marks on being verbose
  - Sometimes, adding an example makes things easier
- ★ The higher the marks are, the more you need to check your answers
- ★ You can answer *any combination of questions as long as the total marks attempted is less than or equal to 25*, i.e., any question attempted beyond 25 marks will not be considered while marking
- ★ If you are making any valid assumption(s) while writing an answer, do remember to mention that information clearly and concisely
- ★ For 1 Mark questions, no explanations are required; just write the answers.
- ★ For 2 mark questions, you can just write the coding logic (2-4 lines), no need to write full code
- ★ For 4 mark questions, write full codes, minor mistakes *(missing a semicolon, forget to close a bracket, etc.)* are ok, major mistakes (*messing up syntax for a loop, switch case, incorrect function prototype, etc.*) are not.
- ★ 2 bonus marks will be awarded if there is *no minor mistakes* as well as *major ones*
- ★ Consider all questions are for C language and assume the size of *int* and *float* as 4 bytes, *char* as 1 byte, *double* as 8 bytes in this exam; also note the characters are evaluated using their ASCII values A-Z are valued 65-90 and a-z are valued 97-122 respectively

---

Q1.1. **Mark 1**

Which of the following are *not valid* variable name(s) in C language?

_name, high5, 4square, do_while, digit-sum, main

Q1.2. **Mark 1**

What will be the value of the variable count at the end of the loop?

```
int i, j, count = 0, n = 20;
for (i = 0; i < n; i++)
    for (j = n - i; j > 0; j = j - 2)
            count++;
```

Mark 1

Q1.3.
What will be the content of the array A, after the following code segment is executed?

```
int A[6] = {3, 5, 2, 4, 9, 1};
int i = 5;
do {
        A[6-i] = A[6-i] + A[5-i];
        i--;
} while (i > 0);
```

Mark 1

Q1.4.
What will be the output of the following code segment for a=20 and for a=0?

```
int a;
switch(a) {
        case 10: a = a + 20;
                 break;
        case 20: a = 10;
        case 37: a * 30;
        default:  a %= 6;
        case 40: a = a + 40;
                 break;
        case 50: a = a + 50;
        }
printf("%d", a);
```

Mark 1

Q1.5.
What will be the value of the variable *count* at the end of the loop?

```
int i = 200, j = 50, count = 10;
while(i > 0 && j > 0)
{
        i = i - j;
        j = j - 5;
        count++;
}
```

Mark 1

Q1.6.
What will be the output of the following code segment?

```
int x = 1, y = 0, z = 1, t;
for (t = 0; t < 10; ++t)
{
        y += (x) ? z : -z;
        z++;
        x = !x;
}
printf("y = %d", y);
```

Q1.7. **Mark 1**

What will be the output of the following code segment?

```
int x = 0;
if (x = 0)
        printf ("if case: %d", x);
else if (x -= 2)
        printf ("1st else-if case: %d", x);
else if (x == -2)
        printf ("2nd else-if case: %d", x);
else
        printf ("else case: %d", x);
```

Q1.8. **Mark 1**

What will be the output of the following code segment?

```
int a[5] = { 15, 1, 5, 30, 25 };
int i, j, k;
i = ++a[1] ;
j = a[1]++ ;
k = a[i++] ;
printf ("%d,%d,%d", i, j, k);
```

Q1.9. **Mark 1**

Write the output of the following code segment.

```
printf ("Expression values = %d %d %f \n", 5/2*2, 2+5/2*3-1, (float) (5/2));
```

Q1.10. **Mark 1**

Write the output of the following code segment.

```
printf ("Expression values = %d %o %x \n", 1024, 1024, 1024);
```

Q2.1. **Mark 2**

Assume that you have taken a character as input (char **ch**) from the user. Your task is to check if **ch** is a lowercase alphabet or not; if yes, then print it, if it is uppercase, then print the lowercase version **ch**, if **ch** is not an alphabet, print the message "*not an alphabet*".

Q2.2. **Mark 2**

Assume that you have taken inputs of principal amount (int **p**), rate of interest in percentage (float **r**) and time of investment in years (int **y**); now you calculate the total amount of interest gathered in simple interest(float **si**) as follows: **si** = **y**/100 * **r** * **p**.

(a) Will this statement above always result in the correct output? If yes, then calculate the value of **si** for **p**=2000, **r**=10.4, **y**=4, and if not, then rewrite the statement correctly.

(b) Also, write the statement that will calculate using **p** and **r**, the time (in years) it takes to double the invested amount.

Mark 2

Q2.3.
Examine the following program carefully and figure out (write in one line) what the program does? Write the output of the program for x = 75 and y = 30

```
#include <stdio.h>
void main()
{
        int x, y ;
        printf("Enter two numbers: " ) ;
        scanf("%d%d", &x, &y);
        while(x!=y)
        {
                if(x>y)  x -= y;
                else     y -= x;
                printf ("x=%d y=%d\n",x,y);
        }
        printf ("%d\n",x);
}
```

Mark 2

Q2.4.
Write a function called *calc_pow* that takes two integers (**x** and **y**) as inputs and returns the value of **x** to the power **y**. Just write the prototype and definition of the function – you must write this code without any major mistakes.

Mark 2

Q2.5.
Figure out the output of g(3) and g(5). The function g(x) is defined below.

```
int g(int x)
{
        int k;
        k = (x>2) ? g(x-1) - g(x-2) : 1;
        return k;
}
```

Mark 2

Q2.6.
Write a function/code segment that will round off a floating point number to two decimal places (e.g. 244.67823 will be rounded off to 244.68).
Hint: To round, you may convert the floating point number appropriately to an integer and then back to a floating point number

Mark 2

Q2.7.
Write a code segment that counts the number of elements not divisible by an integer **k** in a given integer array **A** of size **n**. You may assume that the values of array **A** and the values for **n** and **k** are already scanned from the user. You can write the rest of the logic.

Q3.1. **Mark 4**

I would like to calculate the number of days till a given date from the start of the year of the date.

For example, if the date 13-03-2000 is given, I would like to count 31 days for January 2000, 29 days for February 2000 (since 2000 is a leap year) and 13 days for March 2000 since that date is given; hence, the answer is 31+29+13 = 73 days.

→ Your job is to write the programmatic logic that can achieve this for any user-given date. Show the sequence of execution of your date of birth (as user input).

You may use the following code framework, or you can write your own version.

```
#include<stdio.h>
int main ()
{
int d, m, y, result;
scanf ("%d %d %d", &d, &m, &y); //read the date from the user as values for date, month and year

//verify if the entered data from the user is valid or not i.e. m cannot be greater than 12, y must be non-negative integer, etc.; in the case of invalid data return -1, otherwise, proceed to the next steps

//determine if you need to add an extra day in result

//determine how many days corresponding to the months before the days are to be added

//add the day in the last month

printf ("The count of days is %d", result);
return 0;
}
```

***Hint**: You may use switch case statements to make life easier for you.*

Q3.2. **Mark 4**

I call a non-zero positive integer number *marvelous* if the number is divisible by the sum of its digits.

For example, the number 171 is divisible by (1+7+1) = 9, and so 171 is *marvelous*, whereas 172 is not *marvelous*. I would like to know if a number is *marvelous* or not.

→ Your job is to write the programmatic logic that takes a positive integer number as input and checks whether the number is marvelous or not. Show the sequence of execution of your code for the total mark (given as user input) you obtained in class 12th.

You may use the following code framework, or you can write your own version.

```
#include<stdio.h>
int main ()
{
int n, sum;
scanf ("%d", &n); //read the number from the user

//verify if the entered data from the user is valid or not i.e. n must be a non-zero positive integer; in
the case of an invalid n return -1, otherwise, proceed to the next steps

//initialize the sum

//think and use known tools to extract a digit from n and add it to the sum

//adjust the parameters of your solution and repeat until ...

//check if n satisfies the condition for being marvelous and print appropriate message
return 0;
}
```

***Hint**: You will need to use the modulus operator and looping*

Mark 4

Q3.3.

I have an array of integers in arbitrary order. Had the elements of the array were sorted, some of the elements would be in different positions (indexes) of the array. This (would be) position for some element is known as the rank of that element. I do not want to sort the array, but I still would like to know the rank of all the elements in the array.
For example: I have the array A = { 34, 23, 12, 100 }, then the sorted version of the array would be { 12, 23, 34, 100 }; hence the rank of 34 is 2, rank of 23 is 1, rank of 12 is 0, and rank of 100 is 3.

→ Your job is to write the programmatic logic that prints the rank of all the elements in the array. You may assume that the array A has already been scanned.

You may use the following code framework, or you can write your own version.

```
#include<stdio.h>
#define size 10
int main ()
{
int A[size];
int i;
//scan the array from the user
for (i=0; i<size; i++)
        scanf ("%d", &A[i]);

//for each element in the array

        //calculate the rank of the element

return 0;
}
```

***Hint**: rank of an element is basically is the number of elements present in the array which are smaller than that element*