Computer Programming and Problem Solving in C

Q1

(a) In C programming it do not allow directly to return multiple values from a function. But we can easily achieve this by returning

① Pointers
② Structurs
③ Arrays

Using Structures

As the structure is a user defined data type. We can defin a structure that consiest multiple data in it and we can easily return the structure variable from the function, and multiple values from it.

* Let an example where the functions find the greater and smaller of two distinct number.

```
#include <stdio.h>
typedef struct greaterSmaller {
        int great, small;
};
greaterSmaller findgreaterSmaller (int a, int b)
{
    greaterSmaller s;
```

```
    if (a > b)
    {
      s.great = a;
      s.small = b;
    }
    else
    {
      s.great = b;
      s.small = a;
    }
    return s;
}

int main()
{
    int x, y;
    greaterSmaller res;
    scanf("%d %d", &x, &y);
    res = findGreaterSmaller(x, y);
    printf("\n Great is %d and Smaller is %d",
            res.great, res.small);
    return 0;
}
```

using Array

We can also return multiple values using array but here the returning values should be same type as array

construct with similar datatype.

e.g. with the same privious example.

```
void findGreaterSmaller(int a, int b, int res[])
{
    if (a > b){
        res[0] = a;
        res[1] = b;
    }
    else
    {
        res[0] = b;
        res[1] = a;
    }
}
int main()
{
    int x, y;
    int ans[2];
    scanf("%d %d", &x, &y);
    findGreaterSmaller(x, y, ans);
    printf("Greater %d, Smaller %d", ans[0], ans[1]);

    return 0;
}
```

Name – Debasish Biswas — Endsem
Regno – 2021PGLACA057
sub – CA 3101

Date – 3/1/22
page – 4

Q1

(b)

```
#include <stdio.h>
int main()
{
  int i, n, marks, A, B, C, D;
    A = 0;
    B = 0;
    C = 0;
    D = 0;
    printf("Enter number of students: ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
     printf("\nEnter total marks of student %d: ", i+1);
     scanf("%d", &marks);
     if(marks >= 90 && marks <= 100) A++;
     else if(marks >= 80 && marks <= 89) B++;
     else if(marks >= 60 && marks <= 79) C++;
     else D++;
    }
     printf("\nTotal students with greade A : %d \n", A);
    printf("Total students with grade B : %d \n", B);
    printf("Total students with grade C : %d \n", C);
    printf("Total students with grade D : %d \n", D);
    return 0;
}
```

Output :

```
Enter number of students: 5

Enter total marks of student 1: 99
Enter total marks of student 2 : 88
Enter total marks of student 3 : 50
Enter total marks of student 4 : 77
Enter total marks of student 5 : 90

Total students with grade A: 2
Total students with grade B : 1
Total students with grade C : 1
Total students with grade D : 1
```

Q2

(a)

(b)

```
#include <stdio.h>

float area (float r) // for calculating area
{
    return (22 * r * r) / 7 ;
}
float volume (float r, float h)
{
    return area(r) * h ; //calling area()
}
```

```
int main()
{
    float vol, rad, hig;
    printf("Enter radius: ");
    scanf("%f", &rad);
    printf("\nEnter height: ");
    scanf("%f", &hig);
    vol = volume(rad, hig);
    printf("\nVolume of cylinder is %f", vol);
    return 0;
}
```

Output:

```
Enter radius: 5
Enter height: 3
Volume of cylinder is 235.619445
```

Q2 (a) A multidimentional array can be expressed in terms of an array of pointers. Here the newly defined array will have one less dimensions than the original multidimenssional array.

therefore,

an n-dimentional array can be defined as an (n-1) dimentional array of pointers by

data_type *array[expression 1][expression 2]...[expression n-1]

Let for 2-D array we can write

((a+i)+j)

that represent a[i][j]

Here the inner pointer *(a+i)+j represent the address of a[i][j].

The outter pointer *(*(a+i)+j) represents the value that is stored in the address a[i][j];

where in the code of array of lower dimension (1.D array) an elemen is represent by *(a+i)

that indecates a[i]

contifing of a singhe pointer, representing value stored at the address of a[i]

i.e,

value a[i] it self, {(a+i) represent the address of a[i]}

Q3 a) The parameters declared in the function defination are local to ~~that~~ the scope of that pirticular function. and it mean nothing outside of that function. Thats why the name of the argument used to invoke a function don't conflict with them. And also the formal and actual argument ~~alloat~~ allote diffrent memory location.

Q3 (b) i)

```
int f(int a)
{
...
...
}

char * fun(int (*f)(int))
{
...
...
}
```

ii)

```
float *function(int *a; int *b; int *c)
{
...
...
return ...
}
```

Q3 (c)

i) In main() function ptmax is the integer pointer which hold the address of the largest element of array a ;

ii) function funct returns a integer pointer that consist the address of largest element of an array.

iii) When the function funct accessed the address of the largest element is assigned to ptmax pointer variable.

iv) The purpose of for loop is to iterate through the array and find the maximum element available in the array

v) *ptmax access the values of the largest element in the array, so it will print

" max = 50 "

Q 4 (a) As we know that the size of the int is not specified by the C standards. C only explains the minimum size of int that is 2 byte. by using typedef keyword we can specify int where the size always 4 byte.

e.g
```
#ifdef AVR_32
typedef int int_32
#else
typedef long int_32
#endif
```

By writing this we can specify that in any machine whenever a int data type is created it will take 4 byte space in the memory.

we can also write somting like

```
typedef int int16_t
```

to specify machine to take only 2 byte for an int variable.

4(b) OR

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[1000];
    char newStr[100][10];
    int i, j, ctr = 0;
    printf("Input a string: \n");
    fgets(str1, sizeof str1, stdin);
    j=0;
    for(i=0; i <= (strlen(str1)); i++)
    {
     if(str1[i]==" " || str1[i]=="!" || str1[i]==":"
        || str1[i]==";" || str1[i]=='\0')
     {
        newStr[ctr][j]='\0';
        ctr++;
        j=0;
     }
    else
     {
        newStr[ctr][j] = str1[i];
        j++;
    } }
```

```
    printf("\n After spliting the String: \n");
    for(i=0; i<ctr; i++)
    {
      printf("%s \n", newStr[i]);
    }
    return 0;
}
```

Output

Input a String:

I am Debasish; Live in Darjeeling | West: Bengal

After spliting the String:

I
am
Debasish
Live
in
Darjeeling
west
Bengal

Q5 (a) The significance of the file offset pointer is it specify the position of the read/write pointer in the file, which helps to access the whole file and apply the file handling function at the positions whichever we are interested.

To reset the pointer to the start of the file

```
fseek(fptr, 0, SEEK_SET);
```

at the start of the function.

Here, fptr is a file, point (FILE) and '0' is for the position.

Q5. (b)

```
#include <stdio.h>
struct HDD
{
 int mb, gb, kb;
} H;
struct HDD showsize (int h, int c, int s)
{
 H.mb = (512 * h * c * s) / (1024 * 1024);
 H.gb = H.mb / 1024;
 H.kb = H.mb * 1024;
 return H;
}
int main()
{
 struct HDD H1;
 int h, c, s;
 printf("Enter Number of head of HDD: ");
 scanf("%d", &h);
 printf("Enter Number of cylinder of HDD: ");
 scanf("%d", &c);
 printf("Enter Number of track of HDD: ");
 scanf("%d", &s);
 H1 = showSize(h, c, s);
 printf("\n\n capacity of the HDD in difrent
        forms: \n");
```

```
    printf("GB = %d", H1.gb);
    printf("\n MB = %d", H1.mb);
    printf("\n KB = %d", H1.kb);

    return 0;
}
```

Output :

Enter number of head of HDD : 1 2 3 1
Enter number of cylinder of HDD : 1 2 3
Enter number of track of HDD : 1 3 1

Capacity of the HDD in diffrent forms :

GB=1
MB=1493
KB=1528832