Mid-semester

Operating Systems Workshop (CSE 3541)

Programme: B.Tech. (CSE) **Semester:** 5th
Full marks: 30 **Time:** 2 hour
Date: 21/12/2022 **Section:** E
Name: Dibyasundar Das **Solution for Mid-semester**

---

1. a. Assuming that side and area are type double variables containing the length of one side in cm and the area of a square in square cm, write a statement that will display this information in this form: [2]

```
The area of a square whose side length is ______ cm is ______ square cm.
```

Ans:

```c
#include<stdio.h>
int main()
{
    double length = 12;
    double area = length *length;
    printf("The area of a square whose side length is
            %lf cm is %lf square cm.", length, area);
    return(0);
}
```

Output :

```
The area of a square whose side length is 12.00 cm is 144.00 square cm.
```

b. Write the #define preprocessor directive and declarations for a program that has a constant macro for PI (3.14159) and variables radius, area, and circumf declared as double , variable num_circ as an int , and variable circ_name as a char. [2]

Ans:

```c
#include<stdio.h>
#define PI  3.14159
int main()
{
```

```
    double radius, area, circumf;
    int num_cric;
    char circ_name;
    return(0);
}
```

c. If you have N eggs, then you have N/12 dozen eggs, with N%12 eggs left over. Write a program that asks the user how many eggs she has and then tells the user how many dozen eggs she has and how many extra eggs are left over. A gross of eggs is equal to 144 eggs. Extend your program so that it will tell the user how many gross, how many dozen, and how many left over eggs she has.[2]

Ans:

```
#include<stdio.h>
int main()
{
    int N;
    printf("Eneter the number of egg : ");
    scanf("%d", &N);
    printf("You have %d dozens of egg and %d extra eggs.\n", N/12, N%12);
    printf("You have %d gross of egg and %d extra eggs.\n", N/144, N%144);
    return(0);
}
```

Output :

```
Eneter the number of egg : 294
You have 24 dozens of egg and 6 extra eggs.
You have 2 gross of egg and 6 extra eggs.
```

2. a. Trace the execution sequence and determine the output of the following code snippet. [2]

```
int jumble(int x, int y)
{
    x = 2 * x + y;
    return x;
}
int main()
{
```

```
    int x = 2, y = 5;
    y=jumble(x,y) ;
    x=jumble(y,x);
    printf("%d\n",x);
    return 0;
}
```

Ans:

Output :

```
20
```

b. Write a program for an automatic teller machine that dispenses money. The user should enter the amount desired (a multiple of 10 dollars) and the machine dispenses this amount using the least number of bills. The bills dispensed are 50s, 20s, and 10s. Write a function that determines how many of each kind of bill to dispense. **[2]**

Ans:

```
#include<stdio.h>
int main()
{
    int amount;
    printf("amount desired (a multiple of 10 dollars) : ");
    scanf("%d", &amount);
    printf("50s bills dispensed : %d\n", amount/50);
    amount = amount%50;
    printf("20s bills dispensed : %d\n", amount/20);
    amount = amount%20;
    printf("10s bills dispensed : %d\n", amount/10);
    amount = amount%10;
    return 0;
}
```

Output :

```
amount desired (a multiple of 10 dollars) : 280
50s bills dispensed : 5
20s bills dispensed : 1
10s bills dispensed : 1
```

c. Write a program to process a collection of scores obtained by students of a class of certain strength . Your program should count and print the number of students with Grade A (80 and higher), Grade B(65-79), Grade C(40-64) and Grade F(39 and below) . Ensure that the entered scores must remain in between 0 and 100(inclusive). Test your program on the following data: [2]

-23 567 65 12 89 32 17 45 41 58 60 78 82 88 19 22 70 88 41 89 78 79 72 68 74 59 75 81 44 59 -23 -12

Ans:

```c
#include<stdio.h>
int main()
{
    int score;
    int grade[4]={0};
    while(scanf("%d",&score)!=-1)
    {
        if(score<0 || score >100)
            printf("%d score is wrong skipping...\n",score);
        else if(score>=80)
            grade[0]=grade[0]+1;
        else if(score<=79 && score>=65)
            grade[1]=grade[1]+1;
        else if(score<=64 && score>=40)
            grade[2]=grade[2]+1;
        else if(score<=39)
            grade[3]=grade[3]+1;
    }
    printf("Count of grade A : %d\n", grade[0]);
    printf("Count of grade B : %d\n", grade[1]);
    printf("Count of grade C : %d\n", grade[2]);
    printf("count of grade F : %d\n", grade[3]);
    return(0);
}
```

Output :

```
-23 score is wrong skipping...
567 score is wrong skipping...
```

```
-23 score is wrong skipping...
-12 score is wrong skipping...
Count of grade A : 6
Count of grade B : 9
Count of grade C : 8
count of grade F : 5
```

3. a. An integer n is divisible by 9 if the sum of its digits is divisible by 9. Develop a program to display each digit, starting with the rightmost digit. Your program should also determine whether or not the number is divisible by 9. [2]

Ans:

```c
#include<stdio.h>
int main()
{
    int num, sum=0, digit;
    printf("Enter the number : ");
    scanf("%d", &num);
    printf("Digits from right to left : ");
    while(num>0)
    {
        digit = num%10;
        printf("%d ", digit);
        sum = sum +digit;
        num = num/10;
    }
    printf("\n");
    if(sum%9==0)
        printf("The number %d is divisible by 9\n", num);
    else
        printf("The number %d is not divisible by 9\n", num);
    return(0);
}
```

Output :

```
Enter the number : 1234
Digits from right to left : 4 3 2 1
The number 0 is not divisible by 9
```

b. ill the formal parameters at the symbols, ??, to the functions f1 and f2. Also write the output of the code segment. **[2]**

```
void f1(int ??,int ??)
{   int c;
    c=a; a=b; b=c;
}
void f2(int ??,int ??)
{   int c;
    c=*a; *a=*b; *b=c;
}
int main(void ){
    int a=10,b=20,c=9;
    f1(a,b);
    f2(&b, &c);
    printf("%d",a);
    printf("%d",b);
    printf("%d",c);
}
```

Ans:

```
void f1(int a, int b)
{   int c;
    c=a; a=b; b=c;
}
void f2(int *a,int *b)
{   int c;
    c=*a; *a=*b; *b=c;
}
int main(void ){
    int a=10,b=20,c=9;
    f1(a,b);
    f2(&b, &c);
    printf("%d",a);
    printf("%d",b);
    printf("%d",c);
}
```

Output :

```
a = 10
b = 9
c = 20
```

c. Develop a program to use the idea of multiple calls to a function with input/output parameters to sort 6 integer numbers in ascending order without using any sorting algorithms. The prototype of the function to be used in your program to sort the numbers is given as void arrange(int , *int*); and also draw the data areas of calling function and arragne() function for the first function call arrange(....). **[2]**

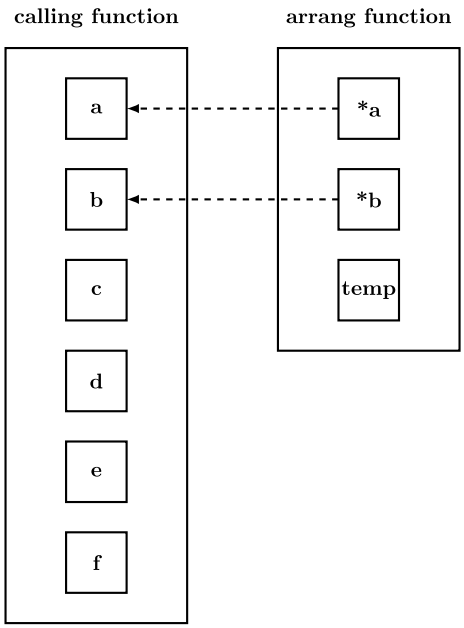
Ans:

```c
#include<stdio.h>
void arrange(int *a, int *b)
{
    int temp;
    if(*a > *b)
    {
        temp = *a;
        *a = *b;
        *b = temp;
    }
}
int main()
{
    int a,b,c,d,e,f;
    printf("Enter 1st ele : "); scanf("%d", &a);
    printf("Enter 2nd ele : "); scanf("%d", &b);
    printf("Enter 3rd ele : "); scanf("%d", &c);
    printf("Enter 4th ele : "); scanf("%d", &d);
    printf("Enter 5th ele : "); scanf("%d", &e);
    printf("Enter 6th ele : "); scanf("%d", &f);

    arrange(&a, &b); arrange(&a, &c); arrange(&a, &d);
    arrange(&a, &e); arrange(&a, &f);

    arrange(&b, &c); arrange(&b, &d); arrange(&b, &e);
    arrange(&b, &f);
```

```
    arrange(&c, &d); arrange(&c, &e); arrange(&c, &f);

    arrange(&d, &e); arrange(&d, &f);

    arrange(&e, &f);

    printf("sorted numbers : %d %d %d %d %d %d\n", a,b,c,d,e,f);
    return(0);
}
```

Output :

```
Enter 1st ele : 2
Enter 2nd ele : 3
Enter 3rd ele : 1
Enter 4th ele : 5
Enter 5th ele : 8
Enter 6th ele : 4
sorted numbers : 1 2 3 4 5 8
```



4 a. Write a program to copy the distinct elements of an int type array to another int type array. For example, if the input array is 4 7 7 3 2 5 5 then the output array will be 4 7 3 2 5. **[2]**

Ans:

```c
#include<stdio.h>
int main()
{
    int arr[]={4,7,7,3,2,5,5};
    int element_count = sizeof(arr)/sizeof(arr[0]);
    int p=0, flag;
    int arr_n[element_count];
    int i,j;
    for(i= 0; i<element_count; i++)
    {
        for(j=0;j<p;j++)
            if(arr[i]==arr[j])
                break;
        if(j==p)
        {
            arr_n[p] = arr[i];
            p = p+1;
        }
    }
    int count_unique =p-1;
    printf("unique array : ");
    for(i=0; i<count_unique; i++)
        printf("%d ", arr_n[i]);
    printf("\n");
    return(0);
}
```

Output :

```
unique array : 4 7 3 2 5
```

b. Given the string char pres[]=“Adams,John Quincy”; and the 40-character temporary variables tmp1 and tmp2. State the execution of the code snippet and output at printf. **[2]**

```c
strncpy(tmp1, &pres[7], 4);
tmp1[4] = '\0';
strcat(tmp1, " ");
strncpy(tmp2, pres, 5);
tmp2[5] = '\0';
```

```
printf("%s\n", strcat(tmp1, tmp2));
```

Output :

```
ohn  Adams
```

c. Organize the Unix commands you have leaned to achieve the given task; (1) create a directory (2) create a file in that directory (3) display the content of the file using cat (4) append few more lines to the file using cat (5) display the list of files. [2]

```
1. mkdir folder
2. echo "Hello !!! How are you" > folder/file.txt
3. cat folder/file.txt
4. cat >> folder/file.txt [enter the lines and press ctrl+D to end]
5. ls folder/
```

5. a. Draw the diagram of the argument array prepared by the shell given to int main(int argc, char *argv[]) for the command line $./a.out "usp dos" SOA iter CSE. [2]

Ans:

	*argv[]
argv[0]	./a.o
argv[1]	usp dos
argv[2]	SOA
argv[3]	iter
argv[4]	CSE
argv[5]	NULL

b.

Write the output of the following code snippet and state the reason for such output. Assume all required headers are present. [2]

```
int main(void) {
    char str[] ="SOA-ITER-IBCS-SUM-IDS-IBCS";
    char ptr[] ="CSE-CSIT-MECH-ECE-EE-EEE";
    char *sptr,*token,*ptoken;
    token=strtok(str,"-");
```

```
    ptoken=strtok_r(ptr,"-",&sptr);
    while (ptoken!=NULL){
        printf("Token=%s---%s\n",token,ptoken);
        token=strtok(str,"-");
        ptoken=strtok_r(NULL,"-",&sptr);
    }
    return 0;
}
```

Ans:

Output :

```
Token=SOA---CSE
Token=SOA---CSIT
Token=SOA---MECH
Token=SOA---ECE
Token=SOA---EE
Token=SOA---EEE
```

c. Construct the process tree diagram with the output of the following code snippet assuming fork never fails. **[2]**

```
int main(void){
int i;pid_t childpid;
    for(i=1;i<8;i++){
        childpid=fork();
        if(childpid<=-1)
            break;
    }
    printf("is-a-parent relationship\n");
    return 0;}
```

Ans:

Output :

```
128 times "is-a-parent relationship"
```