UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 1, Examination in Engineering, Dec 2023

Module No: EE1102 Module Name: Programming Fundamentals

[1 hour and 30 minutes]

[Answer all questions. Questions 1, 2 and 3 carry 8,9 and 8 marks, respectively.]

Part II

Q1. The ordered execution of code can be altered by employing control statements such as **for**, **while**, **do–while**, **if–else if–else**, and **switch**.

a) i. Use a **while** loop to substitute the for loop, expressed as **for**(i=0;i<100;++i) printf("%d",i);.

Answer

```
int i = 0; while (i < 100) { printf(" %d", i); i++; }
```

ii. Explain how the **switch** statement operates, either through an example or a flow chart.

Answer

```
switch (expression)
{
    case value1: // code to be executed if expression equals value1
        break;
    case value2: // code to be executed if expression equals value2
        break;
        // ..
        // more cases as needed

        default: // code to be executed if none of the cases match
}
```

Here's how the switch statement operates:

1. The expression is evaluated once.
2. The value of the expression is compared with each case value.
3. If a match is found, the code block associated with that case is executed.
4. The break statement is used to exit the switch statement. Without break, the control would fall through to subsequent cases (this is called "fall-through").
5. If no match is found, the default case (if present) is executed. The default case is optional.

Example or flow chart

iii. Describe, with an example, how to use an **if−else if−else** statement as a substitute for the code implemented using a **switch** statement.

Answer

```
int day = 3;
if (day == 1) { printf("Monday");}
else if (day == 2) { printf("Tuesday");    }
else if (day == 3) { printf("Wednesday"); }
else if (day == 4) { printf("Thursday"); }
else if (day == 5) { printf("Friday"); }
else { printf("Weekend");  }
```

In this example:

1. The if statement checks if day is equal to 1, and if true, it prints "Monday."
2. The else if statements provide additional conditions to check sequentially. If any of these conditions are true, it skips the subsequent else if and else blocks.
3. The final else block acts as the default case, similar to the default case in the switch statement. If none of the previous conditions is true, it prints "Weekend."
4. This if-else if-else construct achieves the same branching logic as the original switch statement. Each else if block serves as a case, and the final else block acts as the default case.

[6 Marks]

b) The *median* in a data set refers to the central value when the data is organized in ascending order. Its interpretation varies based on the dataset's size: if the number of values is odd, the median is the middle value; if it's even, the median is the average of the two middle values.
If the dataset is represented as an array defined by **float** num[N];, write code snippet to display the median, with N representing a positive integer.

Answer

```
int ar[N] = {5, 10, 15, 20, .. 0, 35, 40, 45, 50}; // Example data set
int temp, median;

for (int i = 0; i < N-1; i++) // // Sort the array
{  for (int j = 0; j < N-i-1; j++)
   {  if (ar[j] > ar[j+1])
      {  temp = ar[j];   ar[j] = ar[j+1];   ar[j+1] = temp;
      }
   }
}

if (N % 2 == 0) // Find the median
       median = (ar[N/2 - 1] + ar[N/2]) / 2;
else   median = ar[N/2];
```

[2 Marks]

Q2. A program to be developed for performing mathematical operations on two complex numbers; $z1 = a + ib$ and $z2 = c + id$. Operations are to be implemented via corresponding functions of given prototypes, as specified below. Each function returns the resulting complex number of the corresponding operation on input complex numbers $z1$ and $z2$. The structure sC represents a complex number.

Mathematical Operation	Function Prototype
$z1 + z2 = (a + c) + i(b + d)$	**struct** sC Add(**struct** sC z1, **struct** sC z2)
$z1 - z2 = (a - c) + i(b - d)$	**struct** sC Sub(**struct** sC z1, **struct** sC z2)
$z1 * z2 = (ac - bd) + i(ad + bc)$	**struct** sC Mul(**struct** sC z1, **struct** sC z2)
$z1/z2 = \frac{(ac + bd)}{(cc + dd)} + i\frac{(bc - ad)}{(cc + dd)}$	**struct** sC Div(**struct** sC z1, **struct** sC z2)

a) i. Define (give implementation of) the structure sC with necessary members to represent a complex number.
Answer **struct** sC { **double** re; **double** im;}

ii. Define the function of prototype **struct** sC Add(**struct** sC z1, **struct** sC z2); for adding $z1$ and $z2$.
Answer

```
struct sC Add(struct sC z1, struct sC z2)
{   struct sC z;
    z.re = z1.re + z2.re;   z.im = z1.im + z2.im;
    return z;
}
```

[4 Marks]

b) i. Define (Implement) the function of prototype **struct** sC setC(**void**);. It allows user to enter a complex number from standard input using scanf(), and then returns the user entered complex number.
Answer

```
struct sC setC()
{   struct sC z;
    printf("Re = ");scanf("%f", &z.re);
    printf("Im = ");scanf("%f", &z.im);
    return z;
}
```

ii. Define the function **int** getMenuItem(**void**); which displays menu,
`a-Add, s-Subtract, m-Multiply, d-Divide, q-Quit`
and returns the user's choice (`a`,`s`, `m`, `d` or `q`). [2 Marks]
Answer

```
int getMenuItem()
{   int key;
    printf("a-Add, s-Substract, m-Multiply, d-Divide, q-Quit");
    scanf("%d", &key);
    return key;
}
```

c) Write the code snippet by using above mentioned functions (assume they are well defined) together with control structures **do**{}**while**(); and **switch**(){} to perform the mathematical operation of user's choice. $z1$ and $z2$ are also entered by the user. After the results of the operation is displayed, menu is displayed again. Choice of `q` ends the program. [2 Marks]

Answer

```
int mItem;
struct z, z1, z2;
do
{
    mItem=getMenuItem();
    switch(mItem)
    {
        case 'a':   z1 = setC(); z1 = setC(); z = Add(z1,z2);
                    printf("Z1+Z2 = %d + i %d", z.re, z.im);
                    break;

        case 's':   z1 = setC(); z1 = setC(); z = Sum(z1,z2);
                    printf("Z1-Z2 = %d + i %d", z.re, z.im);
                    break;

        case 'm':   z1 = setC(); z1 = setC(); z = Mul(z1,z2);
                    printf("Z1 x Z2 = %d + i %d", z.re, z.im);
                    break;

        case 'd':   z1 = setC(); z1 = setC(); z = Div(z1,z2);
                    printf("Z1 / Z2 = %d + i %d", z.re, z.im);
                    break;
    }
}
while(k!='q');
```

d) Define of the function for subtraction if the function prototype is
struct sC * Sub(**struct** sC * z1, **struct** sC *z2); [1 Mark]

Answer

```
struct sC * Sub(struct sC*z1, struct sC * z2)
{   struct sC * z = new struct sC;
    z->re = z1->re + z2->re;
    z->im = z1->im + z2->im;

    return *z;
}
```

Q3. a) The *mode* is identified as the value that occurs most frequently in a data set. For example, in the sequence 99, 86, 87, 88, 111, 86, 54, the mode is 86, occurring twice (highest number of times).
Write a function with the prototype **int** Mode(**int** ar[], **int** N);, that returns the *mode* of data set given in the array ar[ ] of size N. [4 Marks]

Answer

```
int Mode(int ar[], int N)
{    int mode = ar[0]; // Assume the first element as the mode initia
   int maxCount = 0;

   for (int i = 0; i < N; i++)
   {  int count = 1; // Count the frequency of the current element
      for (int j = i + 1; j < N; j++)
      {  if (ar[i] == ar[j])
         count++;
      }
      if (count > maxCount)
      { // If the frequency is greater than the previous maximum
         maxCount = count;
         mode = ar[i]; // Update mode to the current element
      }
   }
   return mode;
}
```

b) The Fibonacci series defined as
$f_n = f_{n-1} + f_{n-2}$
where $n = 1, 2, 3, 4, \ldots$ with initial values $f_1 = 0$, $f_2 = 0$
is the infinite sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,... .
The 1st and 2nd numbers are 0 and 1, respectively. Starting from the 3rd number onward, each subsequent number in the series is the sum of two preceding numbers. E.g. 8th number in the series (13) is equal to the sum of 6th and 7th numbers (5+8) in the series.
Write a function of prototype **unsigned long** nthFibonacci(**unsigned long** n); that returns the n^{th} number in the Fibonacci series. [4 Marks]

Answer

```
unsigned long nthFibonacci(unsigned long n)
{  if (n <= 2) return n - 1; // Base cases for first two Fb numbers

   unsigned long a = 0, b = 1, temp;
   for (unsigned long i = 3; i <= n; i++)
   {  temp = a + b;   a = b;    b = temp;
   }
   return b;
}
```