Type: MCQ

Q1. _____ is a High-speed memory placed between CPU and main memory. (0.5)

1. Read Only Memory
2. Secondary memory
3. **Cache memory
4. Random Access Memory

Q2. Flowcharts and Algorithms are used for (0.5)

1. Better Progarmming
2. Easy testing and debugging
3. Efficient coding
4. **All

Q3. Evaluate the following expression and find the output considering the correct order of precedence. (0.5)

((2^3 - 4) * (5 + 7)) / (6 % 3 + 9/3) + (10 - 2)

1. 12
2. **-4
3. -2
4. 0

Q4. What will be the result of the following expression (0.5)

((7 << 2) >> 1)

1. 6
2. 7
3. **14
4. 28

Q5. What is the output of the error-free C code below (0.5)

```
int main()
{
int x;
x = 2 > 5 != 1 ? 5 < 8 && 8 > 2 ? !5 ? 10 : 20 : 30 : 40;
printf("Value of x: %d",x);
return 0;
}
```

1. 10
2. **20

3. 30
4. 40

Q6. For the error-free C code snippet below, what will be the output? (0.5)

```
int main(){
int i = 4;
switch (i){
    default: ;
    case 3:
       i += 5;
       if ( i == 8) {
       i++;
       if (i == 9) break;
       i *= 2; }
       i -= 4;
       break;
    case 8:
        i += 5;
        break;
 }
 printf("%d",i);
return 0;
}
```

1. 4
2. **5
3. 8
4. 9

Q7. How many times “i” value is checked in this code snippet (0.5)

```c
int main(){
 int i = 0;
 do {
    i++;
    printf("in while loop\n");
 } while (i < 3);
return 0;
}
```

1. 1
2. 2
3. **3
4. 4

Q8. What is the output of the following error-free C code (0.5)

```c
int main(){
 for (int i=0; i<5; i++){
  if (i > 2)
   break;
  printf("PSUC\n");
   }
return 0;
}
```

1. PSUC is printed 2 times.
2. **PSUC is printed 3 times.
3. PSUC is printed 5 times.
4. Compiled Successfully, No Output.

Q9. What is the output of following C program? [Assume int takes 4 bytes] (0.5)

```c
int main(void){
  int i, arr[] = {1, 2, 3, 4, 5};
  for (i = 0; i < sizeof(arr)/arr[4]-1; ++i)
  printf("%d\t", arr[i + 1]);
return 0;
}
```

1. 4 5 9
2. 1 2 3
3. **2 3 4
4. 3 4 5

Q10. What will be the output for the following error-free C code snippet? (0.5)

```
int main(){
 char s[ ]="Midterm Exam";  int first=0,last ; char t;
  for (last=0; s[last]!='\0'; last++);
    while(first<last)
     {last--;
      t =s[first];
      s[first]=s[last];
      s[last]= t;
      first++;}
 printf("%s",s);
return 0;
}
```

1. **maxE mreTdiM
2. mreTmaxE
3. mreTdim
4. mreTdim maxE

Type: DES

Q11. Question 1. (2)

Write a C program to display the positions of all the occurrences of a given character in a character array using linear search. (For input: "Manipal" for a given character 'a', output is: **a found at positions 2, 6**)

```
#include <stdio.h>
int main(){
    char s[50],c;
    int i;
    printf("Enter  the string : ");
    gets(s);
    printf("Enter character to be searched: ");
    scanf("%c",&c); // or c=getchar();
    for(i=0;s[i]!='\0';i++){
     if(s[i]==c){
```

```
      printf("Character '%c' found at location: %d\n ",c,i+1);
     }
    }
 return 0;
}
```

(input : 0.5M, search0.5M, multiple occurances-0.5M, output:0.5M)

Q12. Question 2. (3)

Define precedence and associativity of operators in C. Resolve the below expression stepwise to obtain the final value of x. (Show all intermediate steps)

int x=(-10*(2-3)/15%(5+7)+8*6/12)

Precedence: one operator can have a higher priority, or precedence, over another operator.
Example: * has a higher precedence than +

a + b * c a+(b*c) (0.5)

Associativity: Expressions containing operators of the same precedence are evaluated either from left to right or from right to left, depending on the operator. This is known as the associative property of an operator (0.5)

-10 * (2 – 3) / 15 % (5 + 7) +8*6/12

= -10 * (-1) / 15 % 12+8*6/12 0.5M

=10/ 15 % 12+8*6/12

=0% 12+8*6/12 0.5M

=0+8*6/12

=0+48/12 0.5M

=0+4

=4 0.5M **total 2M**

Q13. Question 3. (3)

Explain the differences between Machine language, Assembly language and High-level languages.

Machine language- The machine language contains only two symbols **1 & 0**. A computer can directly understand the machine language ----1 M
Assembly language- Binary code instructions in machine language are replaced with mnemonics and operands in assembly language. But the computer cannot understand mnemonics, so we use a translator called Assembler to translate mnemonics into machine language. -----1M
High level language with example---- high-level language has a set of predefined words known as Keywords and a set of rules known as Syntax to create instructions. The high-level language is easier to understand for the users but the computer cannot understand it. We use Compiler or interpreter to convert high-level language to machine language.----------1M
Eg: FORTRAN,C, C++, JAVA, Python, etc.,

Q14. Question 4. (3)

Write an algorithm and draw the flowchart to convert a binary to its decimal representation. The algorithm should take a binary number as input and produce its equivalent decimal as output.

Algorithm to Convert Binary to Decimal **1.5M**

Step 1: Start

Step 2: Input a binary number (binary_num) as a string.

Step 3: Initialize a variable decimal_num to 0.

Step 4: Initialize a variable power_of_2 to 1, which represents the current power of 2.

Step 5: Iterate through each character (bit) in the binary_num from right to left (starting with the least significant bit):

BEGIN

a. Convert the current bit character to an integer (0 or 1).

b. Multiply the converted bit by the current power_of_2.

c. Add the result to decimal_num.

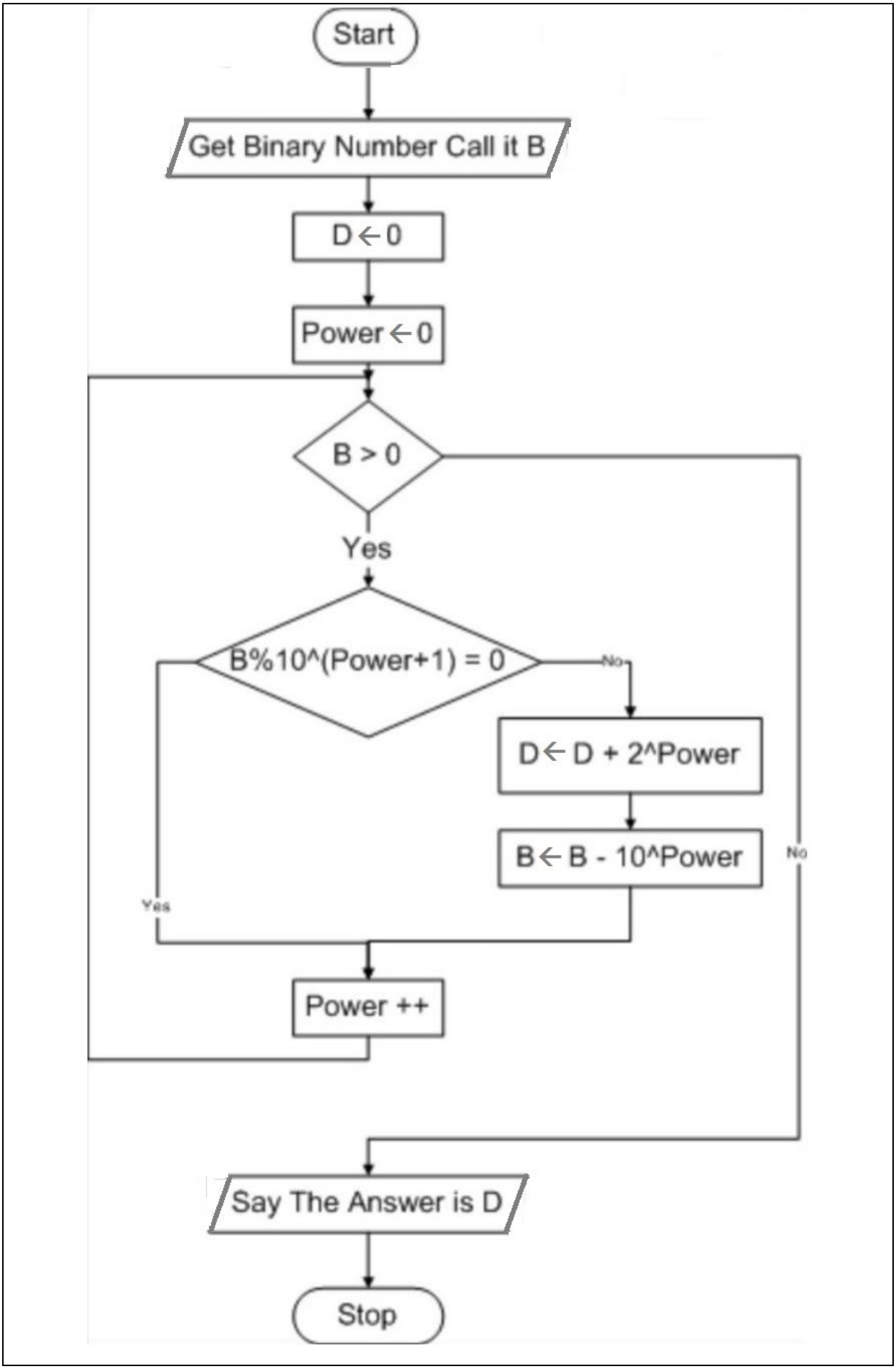
d. Double the value of power_of_2 (multiply it by 2) for the next iteration.

END

Step 6: Display decimal_num as the decimal representation of binary_num.

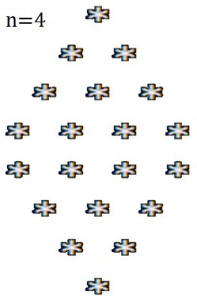
Step 7: End.

Flowchart: 1.5 M


Start
Get Binary Number Call it B
D ← 0
Power ← 0
B > 0
Yes
No
B%10^(Power+1) = 0
No
Yes
D ← D + 2^Power
B ← B - 10^Power
Power ++
Say The Answer is D
Stop

Q15. Question 5. (3)

Write a C program to print a diamond pattern of stars as shown below. [For input n=4 (number of rows])




```
int main()
{
    int rows,column,spaces,i,j,l;
    printf("Enter the number of rows : ");
    scanf(" %d",&rows);
    for(int i=0; i<rows; i++)
    {
        for(l=0;l<rows-1-i;l++)
        {
            printf(" ");
        }
        for(j=0; j<=i;j++)
        {
            printf("%c ",'*');
        }
        printf("\n");
    }
    for(int i=0; i<rows; i++)
    {
        for(spaces=0;spaces<i;spaces++)
        {
            printf(" ");
        }
        for(j=0;j<rows-i;j++)
        {
            printf("%c ",'*');
        }
        printf("\n");
    }

    printf("\n");
    return 0;
}
```

(input: 0.5M, diamond logic(loops usage): 2M, output:0.5M)

Q16. Question 6. (3)

Write a C program to

a) Create an array a[10] and store the following elements in it {2,5,6,7,8,5,4,34,23,4}.
b) Append the code to delete the duplicate elements from the array. Output: Duplicate removed array {2,5,6,7,8,4,34,23}.

```
#include<stdio.h>
int main() {
  int a[10]={ 2,5,6,7,8,5,4,34,23,1}, i, j, k, size=10;//-DECLARING AND INITIALISIMG/Reading-- 1M
  printf("\nArray edited  : ");
// TO  REMOVE THE SIMILAR ELEMENTS------------------------------ 1½ M
  for (i = 0; i < size; i++) {
    for (j = i + 1; j < size;) {
      if (a[j] == a[i])
   {
        for (k = j; k < size; k++) {
          a[k] = a[k + 1];
        }
        size--;
      } else
        j++;
    }
  }

  for (i = 0; i < size; i++) {
    printf("%d ", a[i]);
  }
   printf("Final array is...\n"); //------------------------------ ½  M
   for (i = 0; i < num; i++)
   {
      printf("%d\n", array[i]);
   }

  return (0);
}
```

Q17. Question 7. (4)

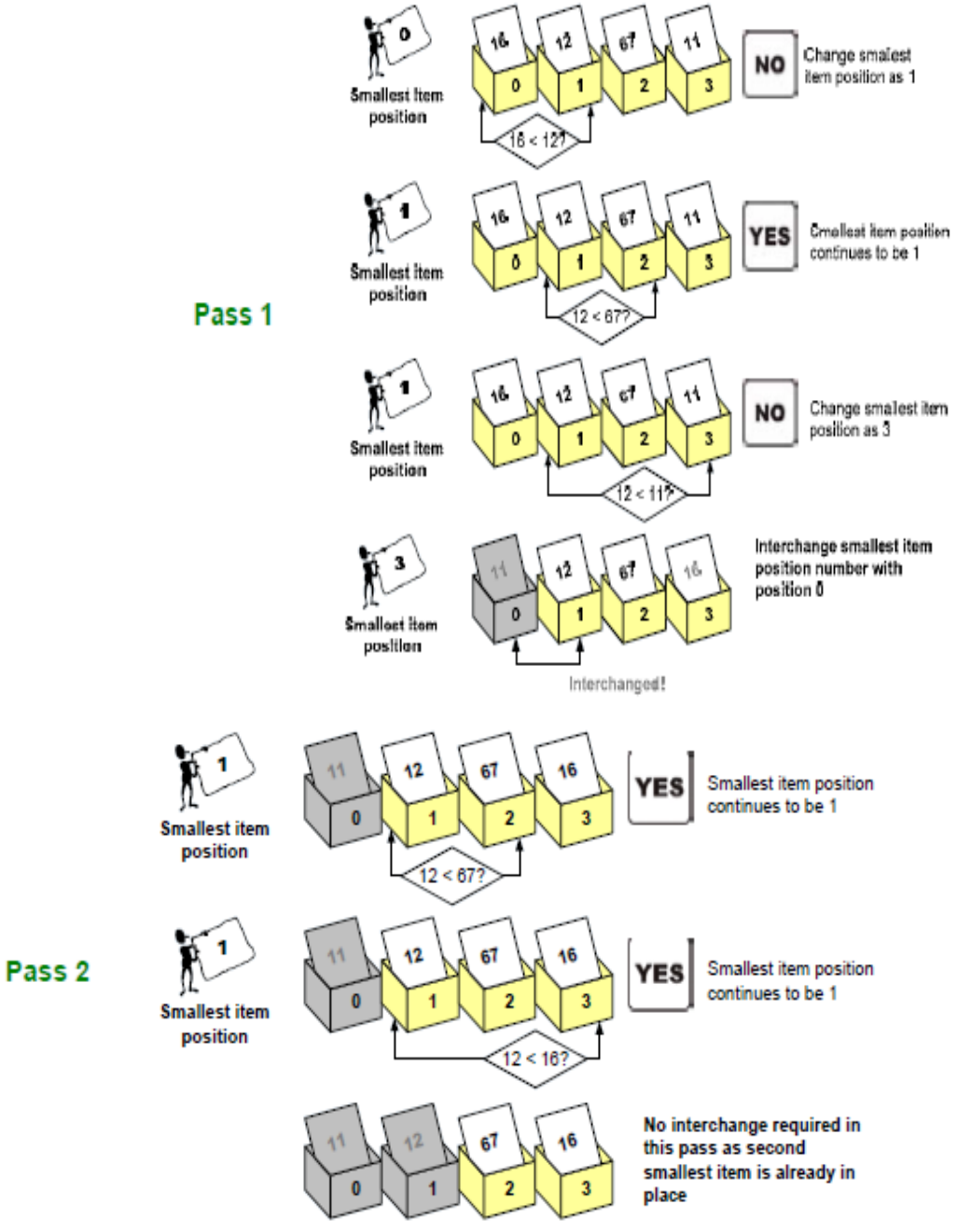
What are the different ways of initializing a 1D array with Zeros? Illustrate the working of the Selection sort algorithm for the given array { 16, 12, 67,11}

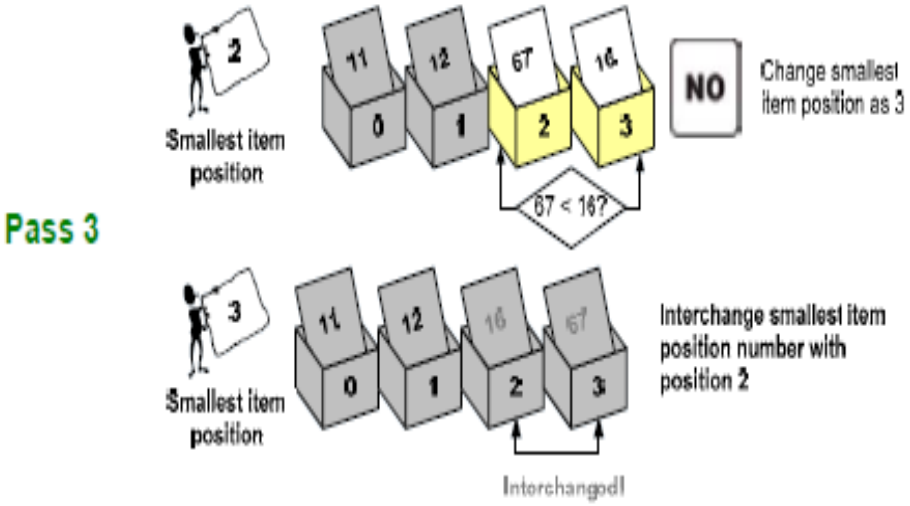
Different ways of initializing 1D arrays:

- ✓ **int number[3] ={0,0,0};**
- ✓ **int number[3] = {0} ;**
- ✓ **int values[20];**

```
for (  i=0;i<3;i++)
        values[i]=0;
```

Selection Sort Illustration:





Initialization of array 1M:
Each Pass : 1M

Q18. Question 8. (4)

Write a C program to find all palindromic prime numbers between limits ***m*** and ***n***, where m & n are greater than 9. [Palindromic prime numbers: 11, 101, 131, 181, …]

```
int main( ) {
   int m, n,i,j,prime,num, dig, rev;
   printf("Enter the values (greater than 10) for LowerLimit & UpperLimit : ");
   scanf("%d %d",&m, &n);
for( i=m; i<=n; i++){
    prime=1;
    num=i;
    for( j=2; j<i; j++ ) {
        if( i % j == 0) {
                    prime=0;
                    break;
       }
     }
 rev=0;
      while(num>0) {
         dig = num % 10;
         rev = rev * 10 + dig;
         num = num / 10;
      }

   if (prime == 1 && rev==i) {
      printf("%d\t",i);
      }
}
   return 0;
}
```