SOFTWARE CPC 1

MAX MARKS: 45

Section A
Basics of programming and OOP
(20 marks)

1) List the types of inheritance supported in C++. (2)
2) What is the difference between encapsulation and abstraction. (2)
3) What is an Abstract Class in C++. (2)
4) What is a destructor? How will you overload a destructor? (2)
5) Out of all basic arithmetic operations (+, -, *, /,) and comparison operations, which are valid on pointers? (2)
6) What is the difference between delete and delete[]?
7) If algorithm A has a time complexity O(n) and algorithm B has a time complexity O(n^2), is it correct to say that A will always take less time to execute than B for identical inputs assuming same programming and execution environment? Explain. (2)
8) Choose the invalid identifier(variable name) from the below: (2)
    a) Int
    b) bool
    c) DOUBLE
    d) __0__

9) Compiler generates ___ file (2)
    a) Executable
    b) Assembly code
    c) Object
    d) Compiler does not create any file

10) How many number of arguments can a destructor of a class receive? (2)
    a) 0
    b) 1
    c) 2
    d) Any number of arguments

Section B
Data structures and algorithms
(25 marks)

Note: Marks will be awarded for correct and efficient code. A well written code with a better time and space complexity will fetch you more marks.

1) Write a function that takes as argument an array and changes it such that:

   a[0]>=a[1]<=a[2]>=a[3]<=a4]..

   If more than one arrangements are possible, change it into any one of them.
   Function prototype is as follows: (5)

```
void change_array(int arr[], int size)
{
        //Definition here
}
```

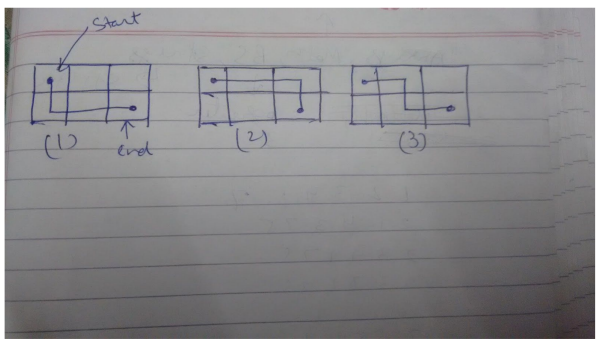
```
Example input array: {4, 1, 5, 3, 2, 7}
Array after change_array(): {5, 2, 7, 1, 4, 3} OR {4, 1, 5, 2, 3} etc.
```

2) Given a grid with of size n * m, write a function that returns the number of distinct paths from the top left (0, 0) cell to bottom right (n - 1, m - 1) cell. At any cell you can either take take a right ((i, j) to (i, j + 1)) or down ((i, j) to (i + 1, j)) as long as you stay inside the grid. It is guaranteed that n and m are such that the answer will fit in a 32 bit integer.
   Function prototype is as follows: (5)

```
int paths(int n, int m)
{
        //Definition here
}
```

```
Example input: 2, 3
Expected return value: 3

Explanation:
The following diagram shows all distinct paths possible (Mad drawing skillz!)
```



3) Write a function to find and return the nth power of an integer a i.e. given a and n, return a^n. Assume a and n are both +ve integers.Function prototype is as follows: (3)

```
int power_func(int a, int n){
        //Definition here
}
```

```
Example input: a = 3, n = 4
Expected return value: 81
```

4) Write a function that takes as input a sorted array of integers and a value v and returns the smallest element greater than v from the array. If no such value is present, return -1. Function prototype is as follows: (5)

```
int find_greater(int arr[], int size, int v)
{
        //Definition here
}
```

Examples:

```
Array = {1, 3, 6, 7, 9}, V = 5
Return value: 6 (6 is the smallest element in the array which is > 5)

Array = {1, 3, 6, 7, 9}, V = 7
Return value: 9 (9 is the smallest element in the array which is > 7)

Array = {1, 3, 6, 7, 9}, V = 10
Return value: -1 (No element in the array is > 10 hence we return -1)
```

5) Write a function that takes as argument two strings A and B and returns 1 if A is present as a substring in B else returns 0. Function prototype is as follows: (7)

```
int string_search(char a[], char b[])
{
        //Definition here
}
```

Examples:

```
A = “this”
B = “this is a sample sentence”
Return value: 1 (A is present as substring in B)

A = “car”
B = “carpenter”
Return value: 1

A = “not present”
B = “sample sentence”
Return value : 0 (A is not present in B)
```

ALL THE BEST :)