NATIONAL UNIVERSITY OF MODERN LANGUAGES, ISLAMABAD
Department of Software Engineering

Final Term Examination SPRING 2021

Name:	***Hamza Mehmood***	**Class/ Section:**	**BS-SE 1ST Evening**
Roll No:	***SP-21-110***		
System ID:	NUML-S21-23529	**Program:**	**BS Software Engineering**
Email:	hamza.mehmood513@gmail.com	**Teacher's Name:**	***Sir Waris Ali***
Course Title:	**Programming Fundamentals**	**Campus:**	**Islamabad**
Course Code:	**SEPF-101**	**DEPT**	**FECS GHAZALI BLOCK**

Submitted to

Sir WARIS ALI

Hamza Mehmood

Sp.21-110

BS

"Question: 3"

(a)

Different ways to use strings in C++ are:

1- String literals:-

"Hello world"

"xyz 123 *&#"

2- C-style strings;

```
char s[20];
```

3- C++ class string:-

```
string s;
```

(b)

Function Overloading:-

Function overloading means that more than one function with the same name exists in the program

but different in the number of arguments. When function is called, then number of arguments will decide that which function will be actually called

Advantages:-

1- Program becomes easy to understand.
2- Easy maintainability of the code.
3- Function overloading brings flexibility in C++ programs.

Example:-

void line(); void line(int);
void line(char); void line(int, char)
void line(char, int)

(d)

Enumerated types (enum) are quite different from struct and unions. An enumerated type is a data type where every possible value is defined as a symbolic constant. Structures do not define lists of constants. A structure cannot contain but an enumeration can not contain structures.

(c)

We can compare pointers if they are pointing to the same array. Relational pointers can be used to compare two pointers. Pointers of the same type compare equal if and only if they are both null, both point to the same function, or both represent the same address.

(Question : 1)

```
#include <iostream>
#include <string>
using namespace std;
void input() {
    int input[500], output[500]
    count, i;
    cout << "Enter number of elements
    in array \n";
    cin >> count;
    cout << "Enter" << count << "number
```

```
for (i=0; i< count; i++){
    cin >> input[i];}}
};

void reverse()
{ int input[500], output[500]
  , count, i;
  cout << "Enter number of elements
    in array \n";
  cin >> count;
  cout << "Enter" << count << "
    numbers \n";
  for (i=0; i<count; i++){
      cin >> input[i];}
  // reversing
    for (i=0; i< count; i++)
  { output[i] = input[count-i-1];
  }
  cout << "Reversed Array in";
  for (i=0; i<count; i++){
      cout << output[i] << " ";
    }}
// largest Array
```

```
void largest array()
{ int i, count;
  float Arr[30];

for (i = 1; i < count; i++)
  { if (arr[0] > arr[i])
      arr[a] = arr[i]; }
  cout << " largest element = "
  << arr[0];

int main(){
  reverse();
  largestarray();
  return 0; }
```

Question 2

```
#include <iostream>
#include <stdlib.h>
#include <math>
using namespace std;

struct distance {
  float feet;
  float inches; };
```

```
distance da;
void input() {
cout << "Enter Radius" << endl;
cin >> da.feet;
cout << "Enter inches" << endl;
cin >> da.inches; }
void fencing() {
double r, area, perimeter;
cout << "Land dimensions" << endl;
input();
area = perimeter * r * r;
perimeter = 2 * Perimeter * r; }
void calculating() {
cout << "cost of fencing and
leveling land" << endl;
cout << length of Total fence: <<
perimeter << endl;
cout << "cost of fencing: << perimeter
*50 << endl;
cout << "Area of circular land:"
<< area << endl;
cout << "cost of leveling" <<
area * 100;
```

```
int main(){
    input();
    bencing();
    calculating();
    return 0;}
```