Explanations
Briefly explain the following concepts:
Question 1 – (4 Marks) Explain the use of keyword '*protected*'.

Question 2– (4 Marks) Name four major principles of object oriented programming.

Debugging
Question 3- (5 Marks) Consider the following code snippet. Would this code compile/run? Justify your answer.
```
#include <iostream>
int main() {
  for (int i = 0; i < 100; i++)
    int *p = new int[i];
  delete[] p;
  return 0;
}
```

Question 4
4 A-(6 Marks) Given the following definition of a Cpu class, define a Laptop class that contains a user-specified number of Cpu's not greater than 10. Include in your definition of the Laptop class:

- **a safe default constructor and**
- **an overloaded constructor that takes as parameters an array of Cpu's and the number of Cpu's.**
- **a query `float calculatePrice() const` - that returns the total price of the total number of cpu's used in the laptop.**

```
class Cpu {
  char model[10];
  float price;
public:
  Cpu() { price = 0; model[0] = '\0'; }
  Cpu(const float _p, const char* _m) { price = _p; strcpy(model, _m); }
  float getPrice() const { return price; }
};
```

4 B -(10 Marks) Code the implementation of the constructors and function of your Laptop class.

Question 5:
5 - (5 marks) Consider the following function, which calculates and returns the sum of the set of values pointed to by x.
```
double sum(const double* x, int n)
{
  double sum = 0.0;
  for (int i = 0; i < n; i++)
    sum += x[i];
  return sum;
}
```
Write a function template that extends this definition to any fundamental type. You may code your solution directly on the above reference code.

Walkthrough

Question 6

6 A- (12 marks) Determine the exact output of the following program:

```
#include <iostream>
class Polygon {
  int num;
protected:
  void setnum(const int _n) { num = _n; }
public:
  Polygon() { num = 0; }
  Polygon(int n) {
    std::cout << "Polygon:" << n << std::endl;
    num = n;
  }
  Polygon(const Polygon& src) {
    num = src.num + 2;
    std::cout << "cc1:" << num << std::endl;
  }
  Polygon & operator=(const Polygon& src) {
    num = src.num - 2;
    std::cout << "=1:" << num << std::endl;
    return *this;
  }
  virtual ~Polygon() {
    std::cout << "~Polygon:" << num << std::endl;
  }
  virtual void go()const {
    std::cout << "Top:" << num << std::endl;
  }
};
class Square : public Polygon {
  int num;
protected:
  int getnum() const { return num; }
public:
  Square(int n) :Polygon(n / 2) {
    std::cout << "Square:" << n << std::endl;
    num = n;
  }
  virtual ~Square() {
    std::cout << "~Square:" << num << std::endl;
  }
  void go()const {
    std::cout << "bottom:" << num << std::endl;
  }
};
void execute(Polygon &p) {
  p.go();
}
```

```
void stop(Polygon p) {
  p.go();
}
int main() {
  Square sq(25);
  std::cout << "--------------" << std::endl;
  execute(sq);
  stop(sq);
  std::cout << "--------------" << std::endl;
}
```

6B- (4 marks) Consider adding the following function calls to the main function in Question 5 A. Which of these calls, if any, would cause a compile or runtime warnings or error?

1) `sq.setnum(4);`
2) `int number = sq.getnum();`