3. Pointers and Things (49%)

Assuming that the following class has been defined:

```
struct pussycat
{ string name;
  string colour;
  int age;
  int number_of_legs; };
```

a.

Write correct C++ declarations for the following things:

i. An array of 1000 pussycat objects, pussycat a[1000];

ii. A pointer to a pussycat object, pussycat * a;

iii. A pointer to an array of pussycat objects, pussycat * a;

iv. An array of 1000 pointers to pussycat objects, pussycat * a[1000];

v. A pointer to an array of pointers to pussycat objects. pussycat * * a;

b.

Draw pictures showing the layout of variables, objects, pointers, and arrays in memory after three objects pussycat objects have been added to the arrays defined in parts iii, iv, and v above. next facing sheet

only 3,4,5 required, but partial for others.

c.

Give correct C++ code for adding the data for the following pussycat

Stumpy, who is grey, eight years old, and has three legs

to the variable that you declared in part v of part a.

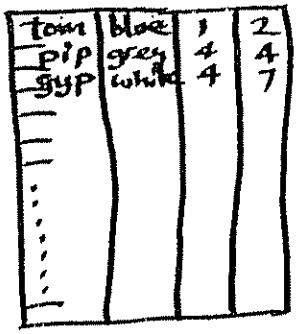
Assume that nothing else has been done. Start with your declaration of "a pointer to an array of pointers to pussycat objects", and show every step required to get the data correctly added. next sheet

d.

Give correct and complete C++ code (no need for a function) that would search through an array as defined for part v, and print the name of the oldest pussycat recorded.

(If you can't do it for the declaration in part v, you can get partial credit for using the declaration from part iv, slightly less for part iii, and some for part i).

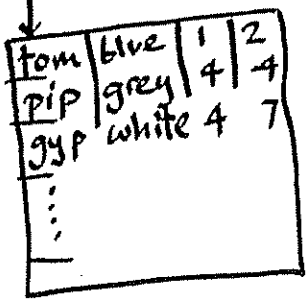
i) a:



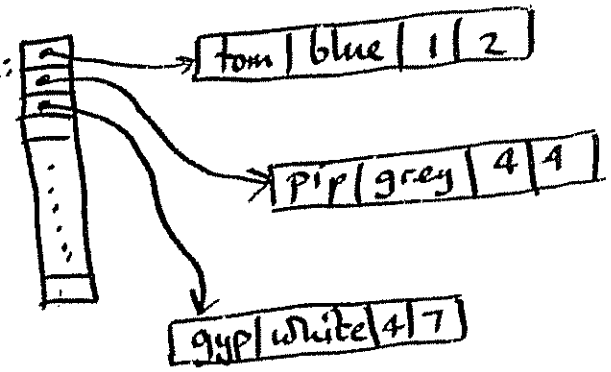
ii) a:

impossible to complete

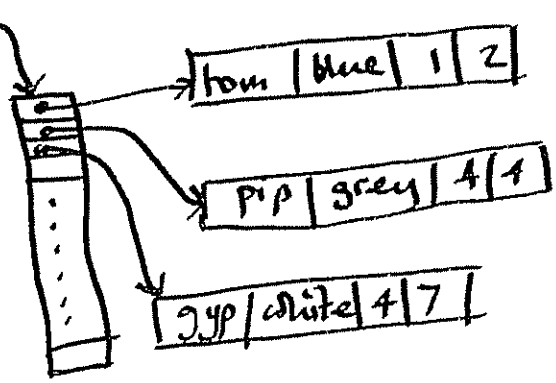
iii) a:



iv) a:



v) a:



(c)

```
pussycat * * a;
a = new pussycat *[10];        // any non-zero value as here
a[0] = new pussycat;
a[0] -> name = "Stumpy";
a[0] -> colour = "grey";
a[0] -> age = 8;
a[0] -> number_of_legs = 3;
```

Or you could add a constructor to the class and use that in place of the last five lines.
Anyone who uses a constructor MUST ALSO define it.

(d)

```
int num = 10; // or however many things there are in the array.
int oldest_age = -1;
String oldest_name = "unknown";
for (int i=0; i<num; i+=1)
    if ( a[i] -> age > oldest_age)
    { oldest_age = a[i] -> age;
      oldest_name = a[i] -> name; }
cout << "The oldest is " << oldest_name << "\n";
```

4. Time and Space (49%)

Assuming that a class or struct called Customer has already been defined, and that the average size of a Customer object is 10,000 bytes, and that the following declarations have been made:

```
Customer * data = new Customer[1];
int number=0, capacity=1;
```

a.

Write a C++ function that will increase the size of the database to a new size that is provided as a parameter. We would expect your function to be able to be used like this: `enlarge_database(25);`

next sheet

b.

A very large number of Customer objects must be added to the database. Each time a new object is to be added, the following statement is executed to ensure that there is enough space for it:

```
if (number>=capacity)
  enlarge_database(number+1);
```

Approximately how many simple operations (copying a Customer object from one location to another) must be performed in total when N (N is a very large number) customers are added to the database

i. with the new size as specified above, `number+1`? $\frac{1}{2}N^2$

ii. if the new size calculation is changed to `number+100`? $\frac{1}{200}N^2$

iii. if the new size calculation is changed to `number*2`? 2N approx

c.

Explain your answer to part iii, show me why it is true.
(If you couldn't do part iii, then explain the last part that you could do.)

next sheet

d.

Assuming that it takes 10^{-10} seconds (one tenth of a nanosecond) to copy a *single byte* from one location to another, calculate the approximate time to insert N items into a database for the three resizing schemes given above, and for these three values of N: one thousand, one million, and one billion. Put your answers into a simple table like this:

How long does it take when	N=1,000	N=1,000,000	N=1000,000,000
new size = old size + 1	half sec.	one week	20,000 years
new size = old size + 100	5 mS	2 hours	200 years
new size = old size × 2	2mS	2 seconds	half hour

working on following sheet.

(a)

```
void enlarge_database(int newsize)
{ // Question said "increase", so no need to
        check that newsize really is larger
  customer * t = new customer[newsize];
  for (int i=0; i<number; i+=1)
      t[i] = data[i];
  delete[] data;
  data=t;
  capacity=newsize; }
```

(c) Capacity starts at 1, and is only ever doubled, so only possible capacities are 1, 2, 4, 8, 16, 32, 64, ... : powers of two

Adding an item always requires one copy operation just to get it in. If the array is not full (i.e. N is not yet a power of two) no extra work is needed.

If the array is full, N == capacity, or N is a power of 2, then all the original N entries must also be copied into a new array, requiring N extra operations

So the work per element is (1 + possibility of N) operations

So total work to get N items in is

$$N + 1 + 2 + 4 + 8 + 16 + 32 + \ldots$$

N: individual items; 1 + 2 + 4 + 8 + 16 + 32 + ...: all the powers of two that are less than N

The sum of a series of powers of 2 is just less than the next power of 2 ($1+2+4+8+16+32 = 63$), which must be close to N itself, ∴ total effort $\approx 2N$ ops.

(d) One op = copy 10,000 bytes @ $\frac{1}{10}$ nS each

→ 1 μS per op.

---

* Size += 1

N items takes $\frac{1}{2}N^2$ operations

$N = 10^3$, $\frac{1}{2}(10^3)^2 = \frac{1}{2} \times 10^6$ ops $\times$ 1 μS = $\frac{1}{2}$ sec

$N = 10^6$, $\frac{1}{2}(10^6)^2 = \frac{1}{2} \times 10^{12}$ ops $\times$ 1 μS = 500,000 sec ≈ 1 week

$N = 10^9$, $\frac{1}{2}(10^9)^2 = \frac{1}{2} \times 10^{18}$ ops $\times$ 1 μS = 500,000,000,000 sec ≈ 20,000 years

IMPORTANT Put times in meaningful units for full marks

* Size += 100

N items takes $\frac{1}{200}N^2$ operations

∴ times are $\frac{1}{100}$ of those above

$N = 10^3 \Rightarrow \frac{1}{200}$ sec or 5mS

$N = 10^6 \Rightarrow$ 5,000 sec or about 2 hours

$N = 10^9 \Rightarrow$ about 200 years

* Size *= 2, N items takes 2N operations

$N = 10^3$, 2×10^3 ops $\times$ 1 μS = 2 mS

$N = 10^6$, 2×10^6 ops $\times$ 1 μS = 2 seconds

$N = 10^9$, 2×10^9 ops $\times$ 1 μS = 2,000 s ≈ $\frac{1}{2}$ hour

5. Special (2½%)

What do you think would happen if I glued two spiders together and called them both Norman? Short answers please.

They would eventually merge to form one giant sixteen-legged schizophrenic super-spider, and I would call it "Normen", or possibly "Lady Jumbarella Ponsonby". It would most probably have psychic powers.

It would not live long enough to enjoy its super-powers because I would be most likely to have used poisonous glue. If not, I would definitely squish such an abomination completely flat with my shoe.