2007 Paper 3 Question 4

Programming in C and C++

A C programmer is working with a little-endian machine with 8 bits in a byte and 4 bytes in a word. The compiler supports unaligned access and uses 1, 2 and 4 bytes to store `char`, `short` and `int` respectively. The programmer writes the following definitions (below right) to access values in main memory (below left):

Address	Byte offset			
	0	1	2	3
0x04	10	00	00	00
0x08	61	72	62	33
0x0c	33	00	00	00
0x10	78	0c	00	00
0x14	08	00	00	00
0x18	01	00	4c	03
0x1c	18	00	00	00

```
int **i=(int **)0x04;

short **pps=(short **)0x1c;

struct i2c {
  int i;
  char *c;
}*p=(struct i2c*)0x10;
```

(*a*) Write down the values for the following C expressions:

```
**i          p->c[2]          &(*pps)[1]          ++p->i
```

[8 marks]

(*b*) Explain why the code shown below, when executed, will print the value 420.

```
#include<stdio.h>

#define init_employee(X,Y) {(X),(Y),wage_emp}
typedef struct Employee Em;
struct Employee {int hours,salary;int (*wage)(Em*);};
int wage_emp(Em *ths) {return ths->hours*ths->salary;}

#define init_manager(X,Y,Z) {(X),(Y),wage_man,(Z)}
typedef struct Manager Mn;
struct Manager {int hours,salary;int (*wage)(Mn*);int bonus;};
int wage_man(Mn *ths){return ths->hours*ths->salary+ths->bonus;}

int main(void) {
  Mn m = init_manager(40,10,20);
  Em *e= (Em *) &m;
  printf("%d\n",e->wage(e));
  return 0;
}
```

[4 marks]

(*c*) Rewrite the C code shown in part (*b*) using C++ primitives and give *four* reasons why your C++ solution is better than the C one. [8 marks]