1. Tablicę *tab*, zdefiniowaną poniżej, umieszczono w pamięci pod adresem 2293584.

```
int tab[5] = {10, 20, 30, 40, 50};
```

Jakie będą wartości następujących wyrażeń (odniesień do tej tablicy):

(a) `*tab` ..... (1 pkt.)

(b) `tab` ..... (1 pkt.)

(c) `tab[1]` ..... (1 pkt.)

(d) `*(tab+4)` ..... (1 pkt.)

(e) `*tab+4` ..... (1 pkt.)

(f) `tab+1` ..... (1 pkt.)

adres	wartość
4206592	4206616
4206596	4206636
4206600	4206656
4206604	4206676
4206608	4206696
...	
4206616	3
4206620	1
...	
4206636	41
4206640	15
4206644	27
...	
4206656	120
4206660	250
4206664	170
...	
4206676	2200
4206680	1100
...	
4206696	5003
4206700	9004
4206704	6007
4206708	8001

2. Tabela obok przedstawia obraz w pamięci zmiennej *p* zdefiniowanej w języku C, jako `int* p[5]`. Każdy wiersz reprezentuje wartość 32-bitową. Jakie będą wartości następujących wyrażeń (odniesień do tej zmiennej):

(a) `**p` ..... (1 pkt.)

(b) `(*p)[1]` ..... (1 pkt.)

(c) `p[0]`

(d) `*p[1]`

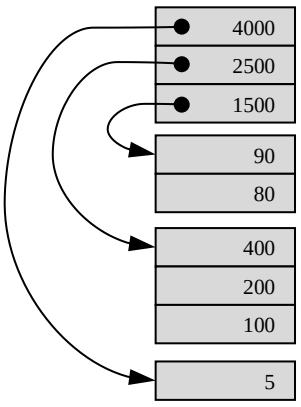
(e) `p[1][1]` ..... (1 pkt.)

(f) `*(p+2)` ..... (1 pkt.)

(g) `*p+2` ..... (1 pkt.)

(h) `*p[2]` ..... (1 pkt.)

3. Rysunek obok przedstawia obraz w pamięci zmiennej *p* zdefiniowanej w języku C, jako `int* p[3]`. Jakie będą wartości następujących wyrażeń (odniesień do tej zmiennej):



(a) `**p` ..... (1 pkt.)

(b) `(*p)[1]` ..... (1 pkt.)

(c) `p[1][1]` ..... (1 pkt.)

(d) `*(p+2)` ..... (1 pkt.)

(e) `*p+2` ..... (1 pkt.)

(f) `*p[2]` ..... (1 pkt.)

4. Jaka będzie wartość zmiennych *a*, *b*, *c*, *d*, *e* i *f* po wykonaniu poniższego fragmentu programu?

```
int tab[5] = {1, 12, 30, 400, 5000};
int *ptr, a, b, c, d, e, f;

ptr = tab + 1;
a = *ptr;
b = ptr[2];
ptr++;
c = *ptr + 2;
d = *(ptr+2);
ptr++;
e = ptr - tab;
f = *ptr - *tab;
```

a ..... (1 pkt.)

b ..... (1 pkt.)

c ..... (1 pkt.)

d ..... (1 pkt.)

e ..... (1 pkt.)

f ..... (1 pkt.)

5. Jaka będzie zawartość tablicy *a* w wyniku wykonania poniższego programu?

```
main(){
   int a[6] = {1, 2, 3, 4, 5, 6};
   int *ptr;

   ptr = a + 2;
   ptr[1] = *ptr + 10;
   ptr[2] = ptr[0] + 20;
   *a = *ptr;
   *ptr = a[1];
}
```

a[0] ..... (1 pkt.)

a[1] ..... (1 pkt.)

a[2] ..... (1 pkt.)

a[3] ..... (1 pkt.)

a[4] ..... (1 pkt.)

a[5] ..... (1 pkt.)

6. Jakie będą wartości zmiennych *x*, *y* i *z* po wykonaniu poniższego fragmentu programu, zakładając, że zmienna *i* została umieszczona pod adresem 2 340 000?

```
int i, *p1, x, y, z;

i = 7;
p1 = &i;
*p1 = 1;
x = i;
y = (int)p1;
z = *p1;
```

x = ... (2 pkt.)

y = ... (2 pkt.)

z = ... (2 pkt.)

7. Jaka będzie wartość zmiennych *a*, *b*, *c*, *d*, *e* i *f* po wykonaniu poniższego fragmentu programu?

```
int *p1, *p2;
int a=1, b=2, c=3, d=4, e=5, f=6;

p1  = &c;
c   = 11;
b   = *p1;
c   = 22;
a   = *p1;
p2  = p1;
e   = *p2;
*p2 = 45;
d   = *p1;
*p1 = 58;
f   = *p2;
```

a ..... (1 pkt.)

b ..... (1 pkt.)

c ..... (1 pkt.)

d ..... (1 pkt.)

e ..... (1 pkt.)

f ..... (1 pkt.)

8. Jaka będzie wartość zmiennych *a*, *b*, *c*, *d*, *e*, *f* po wykonaniu poniższego fragmentu programu?

```
void zmien(int *p, int q){
   static r = 2;
   *p = r;
   r += q;
   q = r;
}

main(){
   int a=0, b=1, c=2, d=3, e=4, f=5;
   zmien(&a, b);
   zmien(&c, d);
   zmien(&e, f);
}
```

a ..... (1 pkt.)

b ..... (1 pkt.)

c ..... (1 pkt.)

d ..... (1 pkt.)

e ..... (1 pkt.)

f ..... (1 pkt.)

9. Jaka będzie wartość zmiennych *a*, *b*, *c*, *d*, *e* i *f* w wyniku wykonania poniższego programu?

```
int ff(int x, int *y){
   x = 10;
   *y = x + 13;
   return x;
}
int gg(int *x, int y){
   return ff(*x, &y);
}
main(){
   int a=1, b=2, c=3, d=4, e=5, f=6;
   c = gg(&a, b);
   d = ff(e, &f);
}
```

a ..... (1 pkt.)

b ..... (1 pkt.)

c ..... (1 pkt.)

d ..... (1 pkt.)

e ..... (1 pkt.)

f ..... (1 pkt.)

10. Jaka będzie zawartość tablicy *a* po wykonaniu poniższego fragmentu programu?

```
void zmien(int *t, int s){
   *t = s;
   s = t[2];
   t[1] = s;
   s = t[4];
   t[3] = s;
}

main(){
   int z = 0, a[6] = {1, 2, 3, 4, 5, 6};
   zmien(a+1, z);
   *a = z;
   exit(0);
}
```

a[0] ..... (1 pkt.)

a[1] ..... (1 pkt.)

a[2] ..... (1 pkt.)

a[3] ..... (1 pkt.)

a[4] ..... (1 pkt.)

a[5] ..... (1 pkt.)

11. Co zostanie wyświetlone na standardowym wyjściu w wyniku wykonania poniższego programu? Jaka będzie wartość zmiennej *x* po wykonaniu funkcji *funkcja*?

```
int funkcja(char *s){
   int i;
   for (i=0; s[i] != 'x'; i++);
   s[i] = 0;
   return i;
}

main(){
   char a[20] = "abcdexfghij";
   int x;

   x = funkcja(a);
   printf("%s", a);
}
```

x = ... (2 pkt.)

Standardowe wyjście: (4 pkt.)

12. Jaka będzie zawartość tablicy *a* po wykonaniu poniższego fragmentu programu, jeśli funkcja `wartosc` zdefiniowana jest następująco: (1 pkt. za każdą odp.)

a)
```
int wartosc (int x) {
   int v = 1;
   v += x;
   return v;
}
```

b)
```
int wartosc (int x) {
   static int v = 1;
   v += x;
   return v;
}
```

```
int wartosc (int x);
main(){
   int a[3];
   a[0] = wartosc(2);
   a[1] = wartosc(3);
   a[2] = wartosc(4);
}
```

a)

a[0] .....

a[1] .....

a[2] .....

b)

a[0] .....

a[1] .....

a[2] .....

13. Ile bajtów pamięci potrzeba na przechowanie zmiennej typu *T*, zdefiniowanego poniżej, przy ułożeniu z dokładnością do: (a) 1 bajta, (b) 2 bajtów, (c) 4 bajtów, (d) 8 bajtów[1].

```
typedef struct {
   union {
      int   x[2];
      float y[3];
   } u;
   double z;
} T;
```

dokładność do 1 bajta ..... (1,5 pkt.)

dokładność do 2 bajtów ..... (1,5 pkt.)

dokładność do 4 bajtów ..... (1,5 pkt.)

dokładność do 8 bajtów ..... (1,5 pkt.)

14. Jakie adresy zostaną podstawione pod zmienne wskaźnikowe *p* i *q* po wykonaniu poniższego fragmentu programu, jeśli zmienna strukturalna *x* została umieszczona w pamięci pod adresem 1000, a dane układane są z dokładnością do: (a) 1 bajta, (b) 2 bajtów, (c) 4 bajtów.

```
struct STR {
   float  a;
   union UN {
      double b;
      int    c;
   }   u;
} x;
void *p, *q;

p = &x.u.b;
q = &x.u.c;
```

zmien. / dokład.	*p*	*q*
do 1 bajta	..... (¾ pkt.)	..... (¾ pkt.)
do 2 bajtów	..... (¾ pkt.)	..... (¾ pkt.)
do 4 bajtów	..... (¾ pkt.)	..... (¾ pkt.)
do 8 bajtów	..... (¾ pkt.)	..... (¾ pkt.)

15. Zamienna unijna *u*, zdefiniowana poniżej, zajmuje obszar pamięci, którego zawartość z dokładnością do bajta przedstawia tabela pod definicją. Co zostanie wyświetlone na standardowym wyjściu w wyniku wywołania funkcji `printf` przy uporządkowaniu grubo- i cienko końcówkowym. (1 pkt. z każdą odpowiedź)

```
union {
   unsigned int   i;
   unsigned short s;
   char a[4];
} u;
```

66
0
0
0

	uporządkowanie grubokońcówkowe	uporządkowanie cienkońcówkowe
`printf("%d", u.i)`	.....	.....
`printf("%d", u.s)`	.....	.....
`printf("%s", u.a)`	.....	.....

Uwaga: litery alfabetu mają kody ASCII z zakresu 65 – 90, a małe z zakresu 97 – 122.

16. Jaka będzie zawartość tablicy *a*, która jest składową zmiennej unijnej *ia*, zdefiniowanej poniżej, w wyniku wykonania przedstawionego programu przy uporządkowaniu grubo- oraz cienkokońcówkowym?

```
main(){
   union U{
      int i;
      char a[4];
   } ia;

   ia.i = 10;
}
```

	grubokońc.	cienkokońc.
`ia.a[0]`	..... (¾ pkt.)	..... (¾ pkt.)
`ia.a[1]`	..... (¾ pkt.)	..... (¾ pkt.)
`ia.a[2]`	..... (¾ pkt.)	..... (¾ pkt.)
`ia.a[3]`	..... (¾ pkt.)	..... (¾ pkt.)

[1] Proszę przyjąć, że wartość typu `char` zajmuje 1 bajt, `short` — 2 bajty, `float` — 3 bajty, `int` — 4 bajty, `double` — 6 bajtów.

17. W tabeli obok przedstawiona została zawartość poszczególnych bajtów pamięci przydzielonej zmiennej *x* (przy ułożeniu z dokładnością do 1 bajta), zdefiniowanej w następujący sposób:

1
2
0
3

```
union U {
   char c;
   int i;
   struct S {
      short int s1;
      short int s2;
   } s;
} x;
```

Co zostanie wyświetlone na standardowym wyjściu w wyniku wykonania funkcji `printf` zgodnie z poniższym wyszczególnieniem przy uporządkowaniu grubo- i cienkokońcówkowym bajtów:

	grubokońcówkowe	cienkokońcówkowe
`printf("%d", x.c)`	..... (1 pkt.)	..... (1 pkt.)
`printf("%d", x.i)`	..... (1 pkt.)	..... (1 pkt.)
`printf("%d", x.s.s2)`	..... (1 pkt.)	..... (1 pkt.)

18. Czy w wyniku wykonania przedstawionego fragmentu programu nastąpi wyciek pamięci i ewentualnie jaka będzie wielkość tego wycieku?

```
char* ptr;
ptr = malloc(13);
ptr = malloc(21);
free(ptr);
```

We wszystkich poniższych zadaniach *napis* jest ciągiem znaków zakończonych zerem.

19. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby wypełniała ona napis przekazany przez parametr `text` kodem ostatniego znaku tego napisu.

```
void last_char ( char text[] ) {

.....

.....

.....

..... }
```

20. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby zwracała ona wskaźnik na początek drugiego słowa w napisie przekazanym przez parametr `text`, przyjmując jako separatory słów znaki spacji i tabulatora.

```
char* second_word ( char text[] ) {

.....

.....

.....

   return ptr;
}
```

21. Proszę uzupełnić poniższą definicję funkcji w celu sprawdzania, czy napis, przekazany jako parametr jest palindromem. Można rozważyć dwa warianty:
    (a) słowo palindromiczne (np. kajak, zakaz) z rozróżnianiem małych i wielkich liter lub ich utożsamianiem,
    (b) zdanie palindromiczne (np. „Zakopane na pokaz”, „A to kanapa pana Kota”, „Tolo ma samolot”, „Ma tarapaty ta para tam”).

```
int palindrom ( char text[] ) {

.....

.....

.....

.....

   return ……;
}
```

22. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby odwracała ona kolejność znaków w drugim słowie napisu przekazanego przez parametr `text`, przyjmując jako separatory słów znaki spacji i tabulatora.

```
void reverse_word ( char text[] ) {

.....

.....

.....

.....

..... }
```

23. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby wyzerowała ona zawartość tablicy opisanej przez parametry `tab` i `size`, nie używając operatora indeksowania.

```
void zero_tab( int tab[], int size) {

.....

.....

.....

..... }
```

24. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby tworzyła ona w dynamicznie zaalokowanym obszarze pamięci zmienną o wartości początkowej takiej jak wartość parametru.

```
double* copy_var( double v) {

……

……

……

   return ptr;
}
```

25. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby tworzyła ona kopię tablicy opisanej przez parametry `tab` i `size` w dynamicznie zaalokowanym obszarze pamięci.

```
float* copy_tab( float tab[], int size) {

……

……

……

……

   return ptr;
}
```

26. Proszę uzupełnić poniższą definicję funkcji w taki sposób, aby zwracała ona za pośrednictwem parametrów wyjściowych `min` i `max` odpowiednio najmniejszą i największą wartość w tablicy opisanej przez parametry `tab` i `size`.

```
void mx_tab( float tab[], int size, …… min, …… max) {

……

……

……

……

……

…… }
```