תרגיל בית 3 – חלק יבש

1. להלן תוכנית המדפיסה את כל סוגי הטיפוסים בשפת C:

```c
#include <stdio.h>  // printf

struct pair_t {
    char first[100];
    size_t second;
};

int main() {
    struct pair_t types[] = {
        { "char", sizeof(char) },
        { "unsigned char", sizeof(unsigned char) },
        { "short", sizeof(short) },
        { "unsigned short", sizeof(unsigned short) },
        { "int", sizeof(int) },
        { "unsigned int", sizeof(unsigned int) },
        { "long", sizeof(long) },
        { "unsigned long", sizeof(unsigned long) },
        { "long long", sizeof(long long) },
        { "unsigned long long", sizeof(unsigned long long) },
        { "float", sizeof(float) },
        { "double", sizeof(double) },
        { "long double", sizeof(long double) }
    };

    for (size_t i = 0; i < sizeof(types) / sizeof(*types); ++i) {
        printf("\"%s\" is %zu bytes.\n", types[i].first, types[i].second);
    }

    return 0;
}
```

תוצאת התוכנית בטרמינל:

```
acamol@Acamol:~/Desktop/HW3$ gcc -std=c99 basictypes.c -o basictypes
acamol@Acamol:~/Desktop/HW3$ ./basictypes
"char" is 1 bytes.
"unsigned char" is 1 bytes.
"short" is 2 bytes.
"unsigned short" is 2 bytes.
"int" is 4 bytes.
"unsigned int" is 4 bytes.
"long" is 8 bytes.
"unsigned long" is 8 bytes.
"long long" is 8 bytes.
"unsigned long long" is 8 bytes.
"float" is 4 bytes.
"double" is 8 bytes.
"long double" is 16 bytes.
```

2. המפרט העדכני ביותר מתוך GitHub.

הכלים בהם משתמשת C++ להגדרת השפה הם:

2.1. תיעוד
2.2. הגדרה מילולית
2.3. דוגמאות

3. במפרט המקושר לעיל ניתן למצוא בעמוד 64 את הציטוט:
"A type cv void is an incomplete type that cannot be completed; such a type has an empty set of values."
בהמשך נכתב: "...הוא [void] משמש כערך חזרה עבור פונקציות ללא ערך חזרה. כל ביטוי יכול להיות מומר לטיפוס void. כל טיפוס ניתן להמיר ל-void cv (explicitly converted). ביטוי מטיפוס void יכול לשמש אך ורק כ-expression statement, כאופרנד של comma expression, כאופרנד שני או שלישי של אופרטור :?, כאופרנד של typeid, noexcept או decltype, כביטוי בהצהרת return של פונקציה המחזירה ערך מסוג void (return statement), כאופרנד של המרה מפורשת לטיפוס void".

הסבר מונחים:
Expression statement – ביטוי משוערך ללא התייחסות לערך החזרה שלו.
Comma expression – אופרטור בינארי
Explicit type conversion – המרה מפורשת של ערך מטיפוס מסוים לערך בטיפוס אחר.
Return statement – ביטוי שמתחיל ב-return ואחריו איזשהו ערך או ביטוי ומהווה את היציאה מהפונקציה.

שאלה 4

<u>שפת C –</u>
<u>טיפוס + ערך:</u>
a. ניתן ליצור טיפוסים בעזרת struct. לדוגמה:

```
struct complex {
	double re;
	double im;
}
```

מגדיר את הערכים המרוכבים והטיפוס struct complex.

b. טיפוס אנונימי :

```
struct {
	double re;
	double im;
}
```

מגדיר טיפוס חדש ללא שם.
ערך אנונימי : לדוגמא הליטרל 5. ערך שמציין את עצמו.

c. ניתן לתת שם נוסף לטיפוס ע"י typedef:

```
Typedef int num;
```

מגדיר את num כך שהוא שקול ל int.

ניתן לתת שם נוסף לערך ע"י enum:

```
Enum (ZERO);
```

מגדיר את ZERO כך שהוא שקול ל 0.

פונקציה :

a. ניתן להגדיר פונקציה בצורה הרגילה, לדוגמא פונקציה בשם foo:

```
Int foo(){ Return 5;}
```

b. אין אפשרות ליצור פונקציות אנונימיות ב-C. דוגמא לשפה שניתן לעשות זאת: ML (בחלק של ML).
c. לא קיים מנגנון כזה ב-C. קיים למשל ב-Pascal.

משתנה :

a. ניתן להגדיר משתנה בצורה הרגילה:

```
Int x = 5;
```

b. קיים. למשל:

```
malloc(sizeof(char))
```

c. לא קיים ב-C.
דוגמא משפה אחרת: C++

```
Int x = 5;
Int& y = x;
```

פרוצדורה :

a. קיים. ניתן להתייחס לפונקציה שמחזירה void כפרוצדורה.
b. אין פרוצדורה אנונימית ב-C.
c. אין שם נוסף לפרוצדורה ב-C. ב-Pascal קיים על ידי alias modifier

כתובת :

a. ניתן ליצור ישות מסוג כתובת ע"י label

```
Label_name:
```

יוצר כתובת בשם Label_name
b. לא ניתן ליצור כתובת אנונימית ב C.
c. לא ניתן לתת שם נוסף לכתובת ב C.

שפת ML –

ערך:

a. ניתן ליצר ערך ע"י ליטרל, לדוגמא:

```
"a"
```

b. ניתן. דוגמה מאחד השקפים:

```
#: a {a=1, b=2};
```

c. מתן שם נוסף לערך:

```
val a = 5;
val b = a;
```

השם של הערך 5 הוא גם a וגם b.

פונקציה :

a. יצירת פונקציה :

```
fun f(n) = n+1;
```

b. פונקציה אנונימית :

```
fn n => n+1;
```

c. שם נוסף לפונקציה, בדומה לערך:

```
fun f(n) = n+1;
```

```
val x = f;
```

<u>משתנה :</u>

לא קיימים משתנים ב ML ולכן לא ניתן ליצור משתנים.

<u>טיפוס :</u>

a. יצירת טיפוס

```
datatype month = Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Nov |Dec;
```

טיפוס חדש בשם month שמכיל את חודשי השנה.

b. לא ניתן ליצור טיפוס אנונימי ב ML. דוגמא לשפה שניתן לעשות זאת: C (בחלק של C).
c. ניתן לתת שם נוסף לטיפוס ע"י type.

```
Type num = int;
```

מגדיר את num כך שהוא שקול ל int.

<u>פרוצדורה :</u>

לא קיימות פרוצדורות ב ML ולכן לא ניתן ליצור ישויות מסוג זה.

<u>כתובת :</u>

אין בשפה.

<u>שפת פסקל –</u>

<u>ערך :</u>

a. יש ליטרלים בסיסיים המוגדרים בשפה. לדוגמה: 1.
b. קיים. למשל "a".
c. קיים על ידי שימור ב-ptr_identifier.

<u>פונקציה :</u>

a. ניתן ליצור פונקציה ע"י שימוש ב function, לדוגמא:

```
Function foo(..)
	Begin
	…
	End
```

יוצר פונקציה בשם foo

b. אין אפשרות ליצור פונקציות אנונימיות בפסקל. דוגמא לשפה שניתן לעשות זאת: ML (בחלק של ML).
c. ניתן בעזרת alias modifier לעשות function overloading

<u>משתנה :</u>

a. ניתן ליצור משתנה ע"י שימוש ב var, לדוגמא:

```
var x:integer;
```

יוצר משתנה בשם x

b. אין אפשרות ליצור משתנה אנונימי בפסקל. ניתן לעשות ב-C.
c. המילה השמורה ref.

<u>טיפוס :</u>

a. ניתן ליצור טיפוס חדש ע"י שימוש ב type, לדוגמא:

```
Type
	Num = type integer;
```

יוצר טיפוס חדש שמכיל ערכי integer אך לא זהה לו

b. אין אפשרות ליצור טיפוס אנונימי בפסקל. דוגמא לשפה שניתן לעשות זאת C (בחלק של C).

c. ניתן לתת שם חדש לטיפוס ע"י שימוש ב type, לדוגמא:

```
Type
        num = Integer;
```

מגדיר את num כך שהוא שקול ל integer

<u>פרוצדורה :</u>

a. ניתן ליצור פרוצדורה ע"י שימוש ב procedure, לדוגמא:

```
procedure foo(..)
        Begin
        …
        End
```

יוצר פרוצדורה בשם foo

b. אין אפשרות ליצור פרוצדורה אנונימית בפסקל.

c. על ידי שימוש ב-alias modifier

<u>כתובת :</u>

a. ניתן ליצור כתובת ע"י label:

```
Label label_name;
```

אין אנונימי ואין שם נוסף