

RESPONSI
PRAKTIKUM KONSEP PEMROGRAMAN
2024

IDENTITAS

Nama : Gibran Maulana Azmi
NIM : L0124100
Kelas : C
Judul Program : C_Responsi1KP_L0124100_GibranMaulanaAzmi_01.c
Deskripsi Program : Program ini adalah program yang menjalankan sebuah game di terminal dimana tujuan kita adalah menebak sebuah kata (Bahasa Indonesia) yang terdiri dari 5 huruf. user akan mendapatkan 7 kesempatan untuk menebak. jika tidak bisa menebak dalam 7 kesempatan itu, user akan kalah. Fitur-fitur yang ada di game ini adalah duplikat dari keyboard, pengecekan huruf dengan simbol, dan interaktif.

Dokumentasi Program

Analisis Kode

- 1. Library yang digunakan untuk program *Source Code***



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <string.h>
5 #include <unistd.h>
```

-`#include <stdio.h>` adalah *library* yang memungkinkan penggunaan fungsi input/output di dalam program. contohnya penggunaan fungsi `printf()` dan `scanf()`.

-`#include <stdlib.h>` adalah *library* untuk memungkinkannya penggunaan fungsi `malloc()` dan `rand()` di dalam program.

-`#include <time.h>` adalah *library* untuk fungsi `time()`, berguna mengatur generator seed untuk fungsi `rand()` menggunakan waktu sekarang.

-`#include <string.h>` berguna untuk memungkinkannya penggunaan fungsi-fungsi *string*. contohnya `strlen()`, `strcmp()`, `strncpy()`.

-`#include <unistd.h>` adalah library untuk memungkinkannya penggunaan fungsi `usleep()` dimana berguna untuk adanya delay saat program di run.

2. Fungsi PemilihanJawaban

```
1 char* PemilihanSesiJawaban () {
2     srand(time(NULL));
3
4     char* listJawaban[] = {
5         "absen", "asbak", "antik", "badai", "badut", "bagus",
6         "bahas", "bahan", "bahwa", "balok", "bantu", "bapak",
7         "baris", "batas", "batik", "beras", "biasa", "bidan",
8         "bisik", "butir", "canda", "catat", "celah", "cemas",
9         "citra", "damai", "datar", "debut", "dekat", "denda",
10        "darah", "dunia", "enzim", "etika", "fakta", "fokus",
11        "gagal", "ganda", "garam", "gelar", "gempa", "goyah",
12        "hadir", "harga", "hasil", "hujan", "iklan", "indah",
13        "jarak", "jenuh", "kabar", "kadal", "kawat", "kenal",
14        "kolam", "kotak", "lihat", "lidah", "macan", "mandi",
15        "maret", "minta", "mobil", "nanti", "nyata", "ombak",
16        "pahat", "pasti", "petir", "pinus", "pulau", "ramah",
17        "rapat", "rehat", "rumah", "sahur", "senam", "siang",
18        "susun", "tanda", "tebal", "tipis"
19    };
20
21    int jumlahListJawaban = sizeof(listJawaban) / sizeof(listJawaban[0]);
22
23    char* sesiJawaban = listJawaban[rand() % jumlahListJawaban];
24
25    return sesiJawaban;
26}
27 }
```

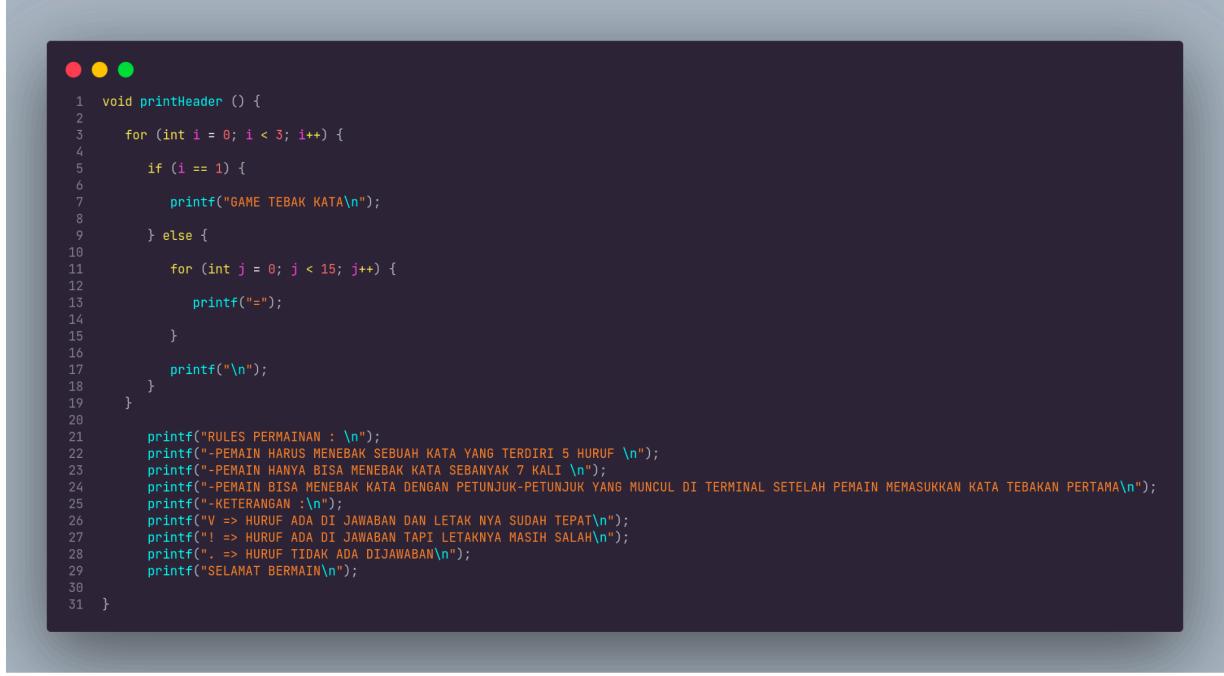
fungsi PemilihanSesiJawaban() bertujuan untuk menghasilkan satu kata secara acak.

fungsi menggunakan `char*` karena fungsi ini akan menghasilkan sebuah jawaban yang bertipe `string`. Semua jawaban akan disimpan di `array` bernama `listJawaban` yang berisi tipe `string`.

cara untuk mengetahui banyaknya jawaban-jawaban di `listJawaban` yaitu dengan cara mengetahui jumlah memori yang digunakan oleh `listJawaban array` menggunakan fungsi `sizeof()` lalu bagi hasil tersebut dengan jumlah memori yang digunakan oleh salah satu element di dalam `listJawaban array` menggunakan `sizeof()` juga. lalu menyimpan hasil itu ke dalam variabel `int` yang bernama `jumlahListJawaban`.

lalu pemilihan jawaban akan disimpan di variabel *string* bernama *sesiJawaban*. Jawaban adalah salah satu element yang ada di *listJawaban array*. Cara agar pemilihan jawaban secara acak adalah dengan menggunakan fungsi *rand()* yang ber-seed oleh *srand(time(NULL))* (seed yang bergantung oleh waktu sekarang), sehingga jika memasukkan *rand()* dimoduluskan(%) oleh *jumlahListJawaban*, hal itu akan men-generate sebuah angka random dari 0 sampai *jumlahListJawaban*. dan jika hal itu dimasukkan ke indeks *listJawaban array*, Jawaban yang diambil sudah terjamin random/acak. Setelah itu, fungsi akan menghasilkan/mengembalikan Jawaban yang ter-generate secara random itu jika fungsi dipanggil.

3. Fungsi *printHeader*



```
1 void printHeader () {
2
3     for (int i = 0; i < 3; i++) {
4
5         if (i == 1) {
6
7             printf("GAME TEBAK KATA\n");
8
9         } else {
10
11             for (int j = 0; j < 15; j++) {
12
13                 printf("=");
14
15             }
16
17             printf("\n");
18         }
19     }
20
21     printf("RULES PERMAINAN : \n");
22     printf("-PEMAIN HARUS MENEBAK SEBUAH KATA YANG TERDIRI 5 HURUF \n");
23     printf("-PEMAIN HANYA BISA MENEBAK KATA SEBANYAK 7 KALI \n");
24     printf("-PEMAIN BISA MENEBAK KATA DENGAN PETUNJUK-PETUNJUK YANG MUNCUL DI TERMINAL SETELAH PEMAIN MEMASUKKAN KATA TEBAKAN PERTAMA\n");
25     printf("-KETERANGAN :\n");
26     printf("V => HURUF ADA DI JAWABAN DAN LETAK NYA SUDAH TEPAT\n");
27     printf("! => HURUF ADA DI JAWABAN TAPI LETAKNYA MASIH SALAH\n");
28     printf(". => HURUF TIDAK ADA DIJAWABAN\n");
29     printf("SELAMAT BERMAIN\n");
30
31 }
```

Tujuan dari *printHeader()* *function* adalah untuk men-*print* header/judul game dan juga keterangan bermain dari game ini. *printHeader()* *function* menggunakan tipe *void* karena fungsi ini tidak akan mengembalikan/menghasilkan sesuatu di dalam program, tetapi hanya mencetak judul dan cara main game ini ke dalam terminal.

metode yang digunakan dalam membuat judul adalah dengan cara *nested loop*, dimana *loop i* akan berulang selama 3 kali (lambang *sama dengan* (=) atas dan bawah ditambah dengan judul game). jika *i* sama dengan 1, perulangan akan mencetak judul game. selain *i* sama dengan 1,

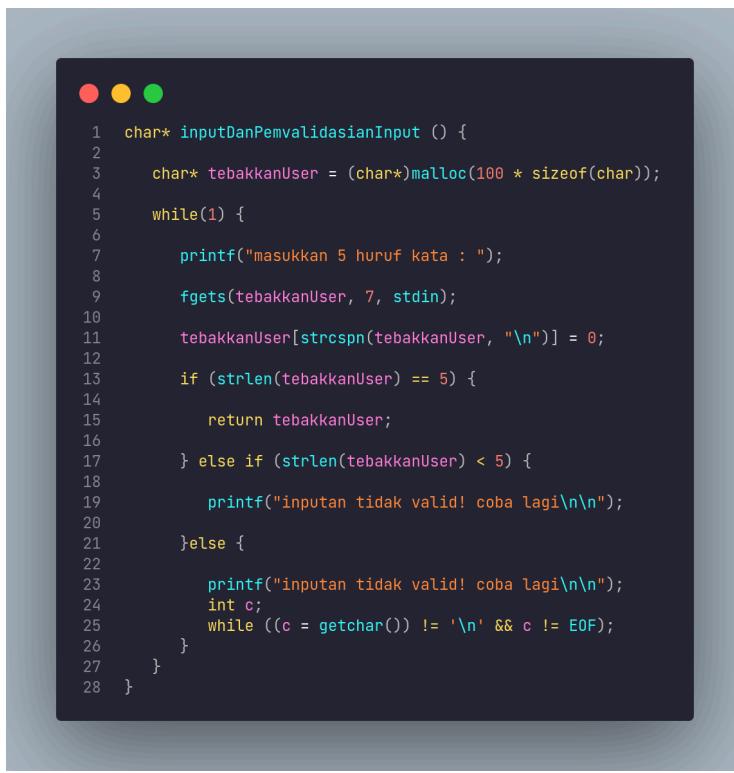
perulangan akan mencetak lambang *sama dengan* (=) yang memanjang menggunakan perulangan *j*, dimana *j* akan berulang sebanyak jumlah huruf (termasuk space) dari judul game.

pencetakan dari keterangan/rules game menggunakan cara manual menggunakan fungsi *printf()*. jika *PrintHeader function* dipanggil, header/judul dan keterangan game akan dicetak ke dalam terminal seperti berikut:



```
● ● ●
=====
1 GAME TEBAK KATA
=====
4 RULES PERMAINAN :
5 -PEMAIN HARUS MENEBAK SBUAH KATA YANG TERDIRI 5 HURUF
6 -PEMAIN HANYA BISA MENEBAK KATA SEBANYAK 7 KALI
7 -PEMAIN BISA MENEBAK KATA DENGAN PETUNJUK-PETUNJUK YANG MUNCUL DI TERMINAL SETELAH PEMAIN MEMASUKKAN KATA TEBAKAN PERTAMA
8 -KETERANGAN :
9 V => HURUF ADA DI JAWABAN DAN LETAK NYA SUDAH TEPAT
10 ! => HURUF ADA DI JAWABAN TAPI LETAKNYA MASIH SALAH
11 . => HURUF TIDAK ADA DIJAWABAN
12 SELAMAT BERMAIN
```

4. Fungsi *inputDanPemvalidasianInput*



```
● ● ●
1 char* inputDanPemvalidasianInput () {
2
3     char* tebakkanUser = (char*)malloc(100 * sizeof(char));
4
5     while(1) {
6
7         printf("masukkan 5 huruf kata : ");
8
9         fgets(tebakkanUser, 7, stdin);
10
11        tebakkanUser[strcspn(tebakkanUser, "\n")] = 0;
12
13        if (strlen(tebakkanUser) == 5) {
14
15            return tebakkanUser;
16
17        } else if (strlen(tebakkanUser) < 5) {
18
19            printf("inputan tidak valid! coba lagi\n\n");
20
21        } else {
22
23            printf("inputan tidak valid! coba lagi\n\n");
24            int c;
25            while ((c = getchar()) != '\n' && c != EOF);
26
27        }
28    }
}
```

inputDanPemvalidasianInput function ini bertujuan untuk menerima input dari user dan juga meng-handle validasi input dari user. jika user memasukkan input yang tidak valid, fungsi ini akan terus mengulang permintaan input kepada user sampai user memasukkan inputan yang valid. jika inputan valid, fungsi akan mengembalikan/menghasilkan inputan dari user ke dalam program. Itulah mengapa tipe dari fungsi ini adalah *char** karena fungsi akan mengembalikan/menghasilkan pointer ke char yang menunjuk kepada inputan dari user yang berupa sebuah string dengan 5 huruf.

cara kerja fungsi ini pertama-tama adalah dengan mendeklarasikan variabel bertipe *char** bernama *tebakkanUser*. bertujuan untuk menyimpan inputan dari user nanti. *malloc* di *assign* ke variabel *tebakkanUser* agar memberi fleksibilitas lebih dalam meng-handle input user, meminimalkan risiko batasan stack, dan memastikan pointer yang dikembalikan valid di luar fungsi. Di dalam kasus ini, *tebakkanUser* dialokasikan dengan *malloc*, pointer tersebut dapat dikembalikan langsung dari fungsi ini. Memori yang dialokasikan dengan *Malloc* tetap tersedia di luar cakupan fungsi (scope) setelah fungsi berakhir, sehingga aman untuk dikembalikan sebagai nilai fungsi. Jika *TebakkanUser* dideklarasikan sebagai array lokal tanpa *Malloc*, memori tersebut akan hilang begitu fungsi berakhir, menyebabkan pointer yang dikembalikan menjadi tidak valid (dangling pointer). *Malloc* mengalokasikan ruang yang cukup besar (100 karakter), meskipun hanya membutuhkan 5 karakter + 1 newline, tetapi memungkinkan fleksibilitas jika panjang input perlu diubah.

lalu, *while(1)* berguna untuk mengulang inputan jika tidak valid. Jika valid, inputan akan dikembalikan/dihasilkan sehingga perulangan otomatis berhenti. penginputan dibaca menggunakan fungsi *fgets()*, dimana akan dibaca hanya 7 element(5 huruf + null terminator(""\0") + newline("\n")) dan akan disimpan di variabel *tebakkanUser*. Lalu, mengecek apakah *newline("\n")* ada di inputan. Dengan cara mencari indeks element *newline* di dalam *tebakkanUser* menggunakan fungsi *strcspn(*tempat yang ingin dicari*, *element yang ingin dicari*)*. jika terdapat *newline* di dalam *tebakkanUser*, maka *strcspn* akan mengembalikan/menghasilkan angka indeks dimana *newline* berada. setelah ketemu posisi *newline*, *tebakkanUser* indeks posisi dari *newline* akan di assign oleh angka 0 untuk menghilangkan element *newline* tersebut.

setelah itu, kita memvalidasi panjang dari string harus 5 menggunakan *if-elseif-else statement*, dengan menggunakan *strlen*(“*string yang ingin di cek panjangnya*”) untuk mengetahui jumlah huruf dari *tebakkanUser*. jika *tebakkanUser* memiliki 5 huruf, *tebakkanUser* langsung dihasilkan/dikembalikan. jika panjang kurang dari 5, program akan meminta inputan ulang dari user.

jika inputan lebih dari 5, buffer dari inputan dibersihkan dulu menggunakan *getchar()* yang dimasukkan ke *while()*. Hal ini mengantisipasi error saat permintaan input ulang. setelah itu program akan meminta user memasukkan input kembali.

5. Fungsi prosesPengecekan

```
1 int prosesPengecekan (char* jawaban, char* tebakkanUser) {
2
3     char pengecekanJawaban[6];
4     strncpy(pengecekanJawaban, jawaban, 5);
5     pengecekanJawaban[5] = '\0';
6
7     char kondisi[] = {'.', '.', '.', '.', '.'};
8
9     for (int i = 0; i < 5; i++) {
10
11         if (pengecekanJawaban[i] == tebakkanUser[i]) {
12
13             kondisi[i] = 'v';
14             pengecekanJawaban[i] = '-';
15         }
16     }
17
18     for (int i = 0; i < 5; i++) {
19
20         if (kondisi[i] != 'v') {
21
22             for (int j = 0; j < 5; j++) {
23
24                 if (tebakkanUser[i] == pengecekanJawaban[j]) {
25
26                     kondisi[i] = '!';
27                     pengecekanJawaban[j] = '-';
28                     break;
29                 }
30             }
31         }
32     }
33
34     for (int i = 0; i < 5; i++) {
35
36         printf("%c", kondisi[i]);
37         fflush(stdout);
38         usleep(500000);
39     }
40
41     printf("\n\n");
42
43     if (strcmp(jawaban, tebakkanUser) == 0) {
44
45         return 1;
46     } else {
47
48         return 0;
49     }
50 }
51 }
```

prosesPengecekan function berfungsi untuk mengecek apakah *jawaban* dan *tebakkanUser* sama atau tidak sama. tipe fungsi ini adalah *int* karena fungsi ini akan

menghasilkan/mengembalikan sebuah 1 (indikasi true jika benar sama) atau 0 (indikasi false jika tidak sama). fungsi ini menerima dua parameter yaitu *jawaban*(yang digenerate oleh fungsi *PemilihanSesiJawaban*) dan *tebakkanUser*(inputan dari user). selain itu fungsi ini mengecek ketepatan huruf per huruf. Fungsi ini akan mencetak apakah setiap huruf berada di posisi yang benar, ada di kata tapi di posisi yang salah, atau tidak ada di kata sama sekali menggunakan simbo-simbol.

char pengecekanJawaban[6] array adalah tempat penyimpanan yang menyimpan 5 elemen + *newline* untuk salinan dari *jawaban* menggunakan *strncpy()*, bertujuan untuk tidak menbandingkan langsung *jawaban* dengan *tebakkanUser* yang biasanya akan error.

char kondisi[] array berguna untuk menyimpan status dari setiap huruf-huruf penebakan. ‘V’ menandakan huruf ada di jawaban dan letaknya sudah benar, ‘!’ menandakan huruf ada di jawaban tetapi letaknya salah, ‘.’ menandakan huruf tidak ada di jawaban. Kondisi secara default adalah ‘.’ karena belum ada pengecekan.

for loop pertama di fungsi ini berguna untuk mengecek apakah huruf per huruf dari *jawaban* dan *pengecekanJawaban* posisinya sudah sama atau tidak. Dengan *if statement*, jika *jawaban[i]* dan *pengecekanJawaban[i]* sama, *kondisi[i]* ditandai dengan ‘V’ dan *pengecekanJawaban[i]* juga ditandai dengan ‘-’ yang bertujuan untuk tidak bertabrakan dalam pengecekan *for loop* kedua.

for loop kedua di fungsi ini menggunakan *nested loop*, yaitu *outer loop* ‘*i*’ dan *inner loop* ‘*j*’ yang bertujuan untuk mengecek apakah salah satu huruf dari *tebakkanUser* ada di *pengecekanJawaban*. *inner loop* hanya akan berjalan jika *kondisi[i]* tidak sama dengan ‘v’, dikarenakan status ‘v’ diprioritaskan. Jadi program akan menskip jika *kondisi[i]* sama dengan ‘v’. Didalam *inner loop*, dengan menggunakan *if statement*, jika *tebakkanUser[i]* sama dengan *pengecekanJawaban[j]* (mengecek setiap huruf *pengecekanJawaban*), *kondisi[i]* ditandai dengan ‘!’ dan langsung di *break* yang berarti huruf ada di jawaban tetapi letaknya masih salah . Dan sama dengan perulangan pertama, *pengecekanJawaban[j]* ditandai dengan ‘-’ untuk menghindari huruf yang sudah di cek agar tidak di cek lagi.

setelah itu, elemen dari *kondisi* di cetak menggunakan *for loop* dengan ada tambahan *fflush(stdout)* dan *usleep(500000)*. *fflush(stdout)* berfungsi untuk mencetak setiap elemen *kondisi* tanpa buffering, dan *usleep(500000)* berfungsi menambahkan jeda selama 0.5 detik setiap cetakan agar game tampak bagus dilihat.

If statement selanjutnya, berfungsi untuk mengecek *jawaban* dan *tebakkanUser* sama atau tidak. Ketika fungsi dipanggil, jika benar, fungsi akan menghasilkan/mengembalikan(*return*) angka 1 (menandakan game sudah selesai). Sebaliknya, jika tidak sama, fungsi akan *return* angka 0 menandakan game masih berlanjut.

6. Fungsi *ngecekCharacterAdaDiString*

```
1 int ngecekCharacterAdaDiString (char hurufDiTebakkan, char* jawaban) {
2
3     int status = 0;
4
5     for (int i = 0; i < 5; i++) {
6
7         if (hurufDiTebakkan == jawaban[i]) {
8
9             status = 1;
10        }
11    }
12
13    return status;
14 }
```

ngecekCharacterAdaDiString function berfungsi membantu fungsi *kondisiKeyboardDanPrintKeyboard* untuk mengecek apakah *hurufDitebakkan* ada di *string jawaban*. Jika benar status akan menjadi angka 1 dan dikembalikan. Jika salah, status akan tetap angka 0 dan dikembalikan. bertipe *int* dikarenakan fungsi akan mengembalikan tipe *integer* saat dipanggil.

7. Fungsi kondisiKeyboardDanPrintKeyboard

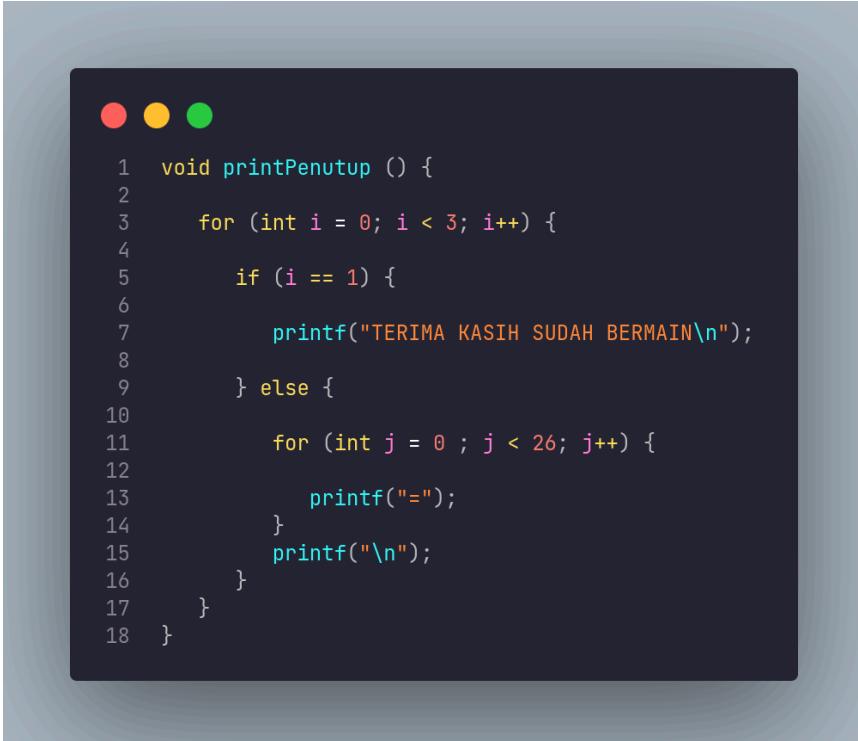
```
1 void kondisiKeyboardDanPrintKeyboard (char* tebakkanUser, char keyboard[3][10], char* jawaban) {
2
3     for (int i = 0; i < strlen(tebakkanUser); i++) {
4
5         char hurufDiTebakkan = tebakkanUser[i];
6
7         for (int j = 0; j < 3; j++) {
8
9             for (int k = 0; k < 10; k++) {
10
11                 if ((ngecekCharacterAdaDiString(hurufDiTebakkan, jawaban) == 0) && hurufDiTebakkan == keyboard[j][k]) {
12
13                     keyboard[j][k] = '_';
14
15                 }
16
17             }
18
19             for (int i = 0; i < 3; i++) {
20
21                 for (int j = 0; j < 10; j++) {
22
23                     if (keyboard[i][j] != '\0') {
24
25                         printf("%c ", keyboard[i][j]);
26
27                     }
28
29                 printf("\n");
30
31             printf("\n");
32
33         }
34     }
```

kondisiKeyboardDanPrintKeyboard function berguna untuk merubah keyboard setiap *tebakkanUser* diinputkan. Fungsi bertipe void dikarenakan fungsi ini tidak mengembalikan nilai apapun. Fungsi ini menerima 3 parameter, yaitu *tebakkanUser*, *keyboard[3][10]*, *jawaban*.

Nested For loop pertama terdiri dari *outer loop* ‘*i*’ dan *inner loop* ‘*j*’. Outer loop akan berulang sebanyak jumlah huruf di *tebakkanUser*, sedangkan inner loop berulang sebanyak 10 kali (banyaknya element setiap *keyboard array*). Fungsi ini berguna untuk mengecek satu-satu huruf dari *tebakkanUser* ada di *string jawaban* dengan memanggil fungsi *ngecekCharacterAdaDiString*. Jika fungsi *ngecekCharacterAdaDiString* mengembalikan angka 0 (huruf tidak ada di *string jawaban*), maka huruf tersebut akan terganti dengan ‘_’ di keyboard.

Nested For loop kedua terdiri dari Outer loop ‘i’ dan Inner loop ‘j’. Outer loop akan berulang sebanyak 3 kali (banyaknya ukuran indeks *row* di *array keyboard*), dan *inner loop* akan berulang sebanyak 10 kali (banyaknya ukuran indeks *column* di *array keyboard*). Berfungsi untuk mencetak semua element terkecuali Null Terminator ('\0') dari *array keyboard*.

8. Fungsi *printPenutup*



```
1 void printPenutup () {
2     for (int i = 0; i < 3; i++) {
3         if (i == 1) {
4             printf("TERIMA KASIH SUDAH BERMAIN\n");
5         } else {
6             for (int j = 0 ; j < 26; j++) {
7                 printf("=");
8             }
9         }
10    }
11 }
```

Fungsi *printPenutup* berfungsi mencetak penutup ketika program sudah selesai ke dalam terminal. Bertipe *void* dikarenakan fungsi ini tidak mengembalikan nilai apapun.

proses pencetakan dari penutup sama dengan pencetakan header/judul.

9. Fungsi *game*

```
1 void game () {
2     char keinginanUserMengulang;
3
4     do {
5         char keyboard [3][10] = {
6             {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'},
7             {'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', '\0'},
8             {'z', 'x', 'c', 'v', 'b', 'n', 'm', '\0', '\0', '\0'}};
9
10    printHeader();
11    char* jawaban = PemilihanSesiJawaban();
12
13    int jumlahTebakkan = 0;
14
15    while(1) {
16
17        char* tebakkanUser = inputDanPemvalidasianInput();
18        printf("%s\n", tebakkanUser);
19
20        if (prosesPengecekan(jawaban, tebakkanUser) == 1) {
21
22            printf("SELAMAT ANDA MENANG || (%s)\n", jawaban);
23            free(tebakkanUser);
24            break;
25        };
26
27        printf("sisa tebakkan (%d)\n", 6 - jumlahTebakkan);
28
29        kondisiKeyboardDanPrintKeyboard(tebakkanUser, keyboard, jawaban);
30        jumlahTebakkan++;
31
32        if (jumlahTebakkan >= 7) {
33
34            printf("YAHH ANDA KALAH || (%s)\n", jawaban);
35            free(tebakkanUser);
36            break;
37        }
38
39        free(tebakkanUser);
40    }
41
42    printf("RESTAR THE GAME?\n");
43    printf("y atau 'Y' untuk MENGULANG game (tekan tombol apa saja selain 'y' atau 'Y' untuk KELUAR dari game) => ");
44    scanf(" %c", &keinginanUserMengulang);
45    int c;
46    while ((c = getchar()) != '\n' && c != EOF);
47
48 } while (keinginanUserMengulang == 'y' || keinginanUserMengulang == 'Y');
49
50 printPenutup();
51 }
```

Fungsi Game berfungsi untuk mengatur jalan nya permainan dan memberikan opsi untuk mengulang permainan. Bertipe *void* dikarenakan fungsi ini tidak mengembalikan nilai apapun.

char keinginanUserMengulang berfungsi untuk menyimpan inputan keinginan user nanti jika user mau mengulang game atau tidak. Di saat akhir game, jika user ingin mengulang game, user bisa menginput huruf ‘y’ atau ‘Y’. Jika ingin keluar dari game, user bisa tekan tombol manapun selain ‘y’ atau ‘Y’.

Kode yang didalam blok kode *do* akan terulang selama user menginput ‘y’ atau ‘Y’ di akhir nanti. Keyboard ini merupakan array 2D yang menyimpan karakter keyboard virtual untuk permainan. Setiap huruf yang telah ditebak akan diubah atau ditandai, jika huruf tidak ada di *jawaban*.

Fungsi *printHeader* akan mencetak judul/Header. Menentukan jawaban bisa dapat di generate dengan memanggil fungsi *inputDanPemvalidasianInput* dan di assign ke variabel *char* jawaban*.

Int jumlahTebakkan berguna untuk men-track berapa kali tebukkan dibuat.

kode blok di dalam *While(1)* akan terus berjalan sampai user menebak *jawaban* atau *jumlahTebakkan* sudah sampai 7. Di dalam kode blok *while* ini user menginput tebakkannya dengan cara memanggil fungsi *inputDanPemvalidasianInput* dan nilai yang dikembalikan fungsi tersebut di assign ke variabel *char* tebukkanUser*.

Setelah itu input akan di cek apakah *tebukkanUser* sama dengan *jawaban* menggunakan nilai yang dikembalikan oleh fungsi *prosesPengecekan*. Jika hasil tersebut sama dengan 1, berarti user sudah menebak katanya dan user menang, jika hasil tersebut 0, user masih harus menebak sampai 7 kali.

printf("sisa tebukkan (%d)\n", 6 - jumlahTebakkan); berguna untuk mencetak sisa tebukkan tersisa.

kondisiKeyboardDanPrintKeyboard(tebukkanUser, Keyboard, jawaban); berguna untuk mendisplay kondisi keyboard sekarang setelah user menginput tebukkan.

jumlahTebakkan++ menambahkan variabel *jumlahTebakkan* untuk mentrack setiap user menebak sampai 7 kali.

If statement (if (jumlahTebakkan >= 7) mengecek apakah pada perulangan saat ini, jumlahTebakkan mencapai sudah 7. Jika sudah mencapai 7 dan masih belum tertebak, user akan kalah. Jika belum sampai 7, user masih ada kesempatan untuk memasukkan inputan.

setiap perulangan memori dari tebakkanUser di bebaskan untuk mencegah kebocoran memori menggunakan free(tebakkanUser).

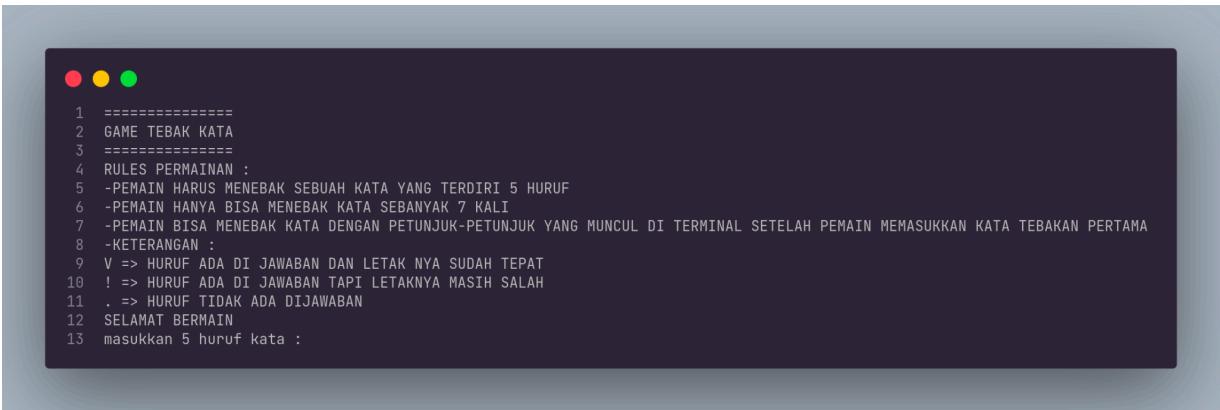
Setelah jumlahTebakkan mencapai 7 atau user sudah menebak katanya, program akan menanyakan user apakah user ingin mengulang game tersebut atau tidak. User akan diperintahkan untuk memasukkan input menggunakan scanf(). jika user menginputkan ‘y’ atau ‘Y’, program akan terulang. Jika user menginputkan selain dari ‘y’ atau ‘Y’, program akan berhenti. Setelah user memasukkan input menggunakan scanf(), buffer akan dibersihkan menggunakan getchar() dan memasukkannya ke while().

Ketika user ingin menutup game, fungsi printPenutup() akan mencetak judul penutup.

Di dalam **Int main**, fungsi *game* dipanggil untuk memulai dan menjalankan program game ini.

Analisis Program (saat dijalankan)

Saat program dijalankan, berikut adalah yang terjadi :



```
1 =====
2 GAME TEBAK KATA
3 =====
4 RULES PERMAINAN :
5 -PEMAIN HARUS MENEBAK SEBUAH KATA YANG TERDIRI 5 HURUF
6 -PEMAIN HANYA BISA MENEBAK KATA SEBANYAK 7 KALI
7 -PEMAIN BISA MENEBAK KATA DENGAN PETUNJUK-PETUNJUK YANG MUNCUL DI TERMINAL SETELAH PEMAIN MEMASUKKAN KATA TEBAKAN PERTAMA
8 -KETERANGAN :
9 V => HURUF ADA DI JAWABAN DAN LETAK NYA SUDAH TEPAT
10 ! => HURUF ADA DI JAWABAN TAPI LETAKNYA MASIH SALAH
11 . => HURUF TIDAK ADA DIJAWABAN
12 SELAMAT BERMAIN
13 masukkan 5 huruf kata :
```

Di line 13, user bisa menginputkan sebuah 5 huruf kata contohnya “kolam”. Jika diinputkan, ini yang akan terjadi :



```
1 masukkan 5 huruf kata : kolam
2 kolam
3 ...V.
4
5 sisa tebakkan (6)
6 q w e r t y u i _ p
7 a s d f g h j _ _
8 z x c v b n _
9
10 masukkan 5 huruf kata
```

Di line 1, adalah inputan user yaitu “kolam”. Di line 2, kolan di cetak lagi karena fungsi *printf* di fungsi *game* berguna untuk membandingkan dengan status dibawahnya. Di line 3, adalah kondisi setiap huruf tepat diatas nya. Bisa dilihat ‘k’, ‘o’, ‘l’, ‘m’ berstatus ‘.’, menandakan huruf tersebut tidak ada di *jawaban*. sedangkan huruf ‘a’ berstatus ‘V’, yang menandakan huruf tersebut ada di jawaban dan letaknya sudah tepat. Di line 5, adalah sisa tebakkan user yang di cetak oleh *printf* yang ada di fungsi *game* (*printf*(“*sisa tebakkan (%d)*\n”, 6 - *jumlahTebakkan*)). Di line 6 - 8 adalah display keyboard dengan cara memanggil fungsi *kondisiKeyboardDanPrintKeyboard*, dimana bisa dilihat huruf yang berstatus ‘.’, didalam keyboard akan tergantikan dengan ‘_’. setelah itu inputan akan terulang.



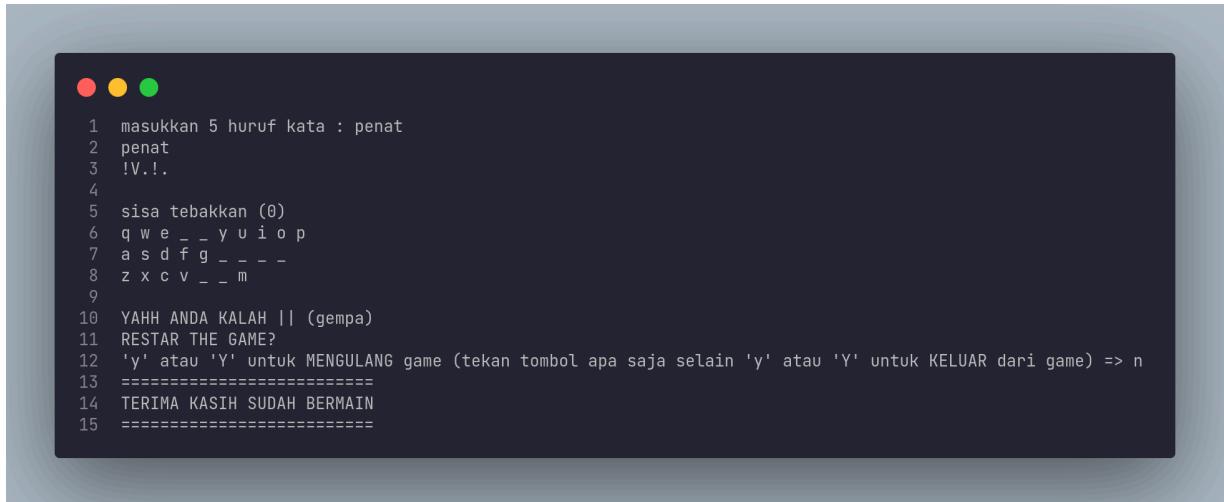
```
1 masukkan 5 huruf kata : badan
2 badan
3 .V!V.
4
5 sisa tebakkan (2)
6 q w _ r t y _ _ _ p
7 a s d f g _ _ _ -
8 z x c v _ _ -
```

Dalam kasus ini, huruf ‘d’ berstatus ‘!’. Itu menunjukkan huruf ‘d’ ada di jawaban, tetapi letaknya masih salah. Jika berstatus ‘!’, di keyboard tidak akan terganti dengan ‘_’.



```
1 masukkan 5 huruf kata : pahat
2 pahat
3 VVVVV
4
5 SELAMAT ANDA MENANG !! (pahat)
6 RESTAR THE GAME?
7 'y' atau 'Y' untuk MENGULANG game (tekan tombol apa saja selain 'y' atau 'Y' untuk KELUAR dari game) => y
8 =====
9 GAME TEBAK KATA
10 =====
```

Dalam kasus ini, user menebak jawabannya. Sehingga program menanyakan user apakah mau mengulang game atau tidak, dalam kasus ini user memasukkan ‘y’, sehingga game mulai kembali.



The screenshot shows a terminal window with a dark background and light-colored text. At the top left, there are three colored dots: red, yellow, and green. The text in the terminal is as follows:

```
1 masukkan 5 huruf kata : penat
2 penat
3 !V!.
4
5 sisa tebakan (0)
6 q w e _ _ y u i o p
7 a s d f g _ _ - -
8 z x c v _ _ m
9
10 YAHH ANDA KALAH || (gempa)
11 RESTAR THE GAME?
12 'y' atau 'Y' untuk MENGULANG game (tekan tombol apa saja selain 'y' atau 'Y' untuk KELUAR dari game) => n
13 =====
14 TERIMA KASIH SUDAH BERMAIN
15 =====
```

Dalam Kasus ini, user gagal menebak *jawaban*. Lalu, program menanyakan user apakah mau mengulang game nya atau tidak. Dalam kasus ini, user memasukkan input selain dari ‘y’ atau ‘Y’. Oleh karena itu, program berakhir.

Tabel penggunaan struktur data :

No.	MATERI	POIN	PERAN DALAM PROGRAM
1	Tipe Data	5	<p>- fungsi yang menggunakan tipe char*</p> <pre> 1 char* PemilihanSesiJawaban () 1 char* inputDanPemvalidasianInput () </pre> <p>Fungsi-fungsi tersebut menggunakan char* karena fungsi tersebut mengembalikan sebuah pointer ke char, yang menunjuk kepada string yang dihasilkan oleh isi dari fungsi-fungsi tersebut.</p> <ul style="list-style-type: none"> • fungsi PemilihanSesiJawaban() akan mengembalikan pointer ke char, yang menunjuk ke ke alamat memori dari string yang dipilih dari array listJawaban • fungsi inputDanPemvalidasianInput() akan mengembalikan pointer ke char, yang menunjuk ke alamat memori dari string yang disimpan dari variabel <i>tebakkanUser</i>. <p>- fungsi yang menggunakan tipe data int</p>

```
1 int prosesPengecekan (char* jawaban, char* tebakkanUser)
```

```
1 int ngecekCharacterAdaDiString (char hurufDiTebakkan, char* jawaban)
```

fungsi-fungsi tersebut menggunakan tipe data **int** karena fungsi tersebut akan mengembalikan sebuah angka (integer) jika fungsi dipanggil.

- fungsi **prosesPengecekan** jika dipanggil akan mengembalikan angka 1 jika nilai *jawaban* dan *tebakkanUser*, jika tidak sama, maka akan mengembalikan angka 0.
- fungsi **ngecekCharacterAdaString** jika dipanggil akan mengembalikan angka 1 jika nilai *hurufDiTebakkan* sama dengan *jawaban[i]*. Jika tidak sama, maka akan mengembalikan angka 0.

- Fungsi yang menggunakan tipe data **void**

```
1 void printHeader ()
```

```
1 void kondisiKeyboardDanPrintKeyboard (char* tebakkanUser, char keyboard[3][10], char* jawaban)
```

```
1  void printPenutup ()
```

```
1  void game ()
```

fungsi-fungsi tersebut menggunakan tipe data **void** karena fungsi tersebut tidak mengembalikan nilai apapun jika dipanggil. Hanya menjalankan instruksi dari kodeblok didalam fungsi.

- Variabel yang menggunakan tipe data **int**

```
1  int jumlahListJawaban = sizeof(listJawaban) / sizeof(listJawaban[0]);
```

```
1  int status = 0;
```

```
● ● ●  
1 int jumlahTebakkan = 0;
```

variabel-variabel tersebut menggunakan tipe data **int** dikarena variabel tersebut menyimpan data/element yang berupa *integer* (bilangan bulat)

-variabel yang menggunakan tipe data **char**

```
● ● ●  
1 char hurufDiTebakkan = tebakkanUser[i];
```

```
● ● ●  
1 char keinginanUserMengulang;
```

variabel tersebut menggunakan tipe data **char** karena variabel tersebut akan menyimpan data/element bertipe data *char* (huruf).

- Variabel yang menggunakan tipe data **char***

```
1 char* sesiJawaban = listJawaban[rand() % jumlahListJawaban]
```

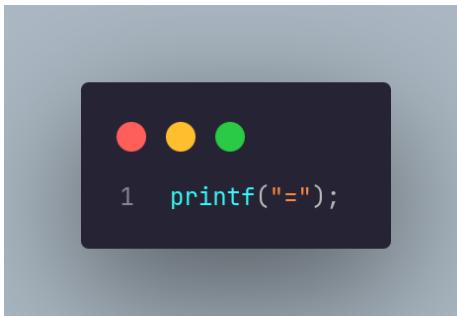
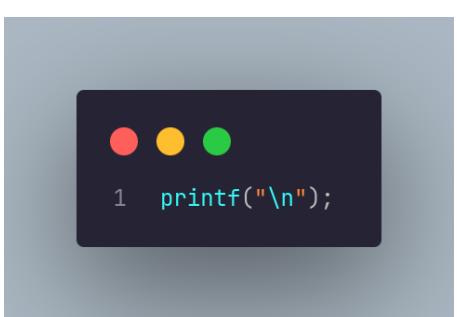
```
1 char* tebakanUser = (char*)malloc(100 * sizeof(char));
```

```
1 char* jawaban = PemilihanSesiJawaban();
```

```
1 char* tebakanUser = inputDanPemvalidasianInput();
```

bertipe *char** karena variabel tersebut akan menyimpan sebuah element string.

- variabel yang menggunakan tipe data **char[]**

			 <pre>1 char pengecekanJawaban[6];</pre>
2	I/O	10	<p>-Output</p>  <pre>1 printf("GAME TEBAK KATA\n");</pre>  <pre>1 printf("=);</pre>  <pre>1 printf("\n");</pre>

```
1 printf("RULES PERMAINAN :\n");
2 printf("- PEMAIN HARUS MENEBAK SEBUAH KATA YANG TERDIRI 5 HURUF \n");
3 printf("- PEMAIN DIBERI 7 KALI UNTUK MENEBAK KATA SEBANYAK 7 KALI \n");
4 printf("- PEMAIN BISA MENEBAK KATA DENGAN PETUNJUK-PETUNJUK YANG MUNCUL DI TERMINAL SETELAH PEMAIN MEMASUKKAN KATA TEBAKAN PERTAMA\n");
5 printf("- KETERANGAN :\n");
6 printf(" => HURUF ADA DI JAWABAN DAN LETAK NYA SUDAH TEPAT\n");
7 printf(" => HURUF ADA DI JAWABAN TAPI LETAKNYA MASIH SALAH\n");
8 printf(" => HURUF TIDAK ADA DI JAWABAN ATAU SUDAH DIAKAGUIN \n");
9 printf("SELAMAT BERMAIN\n");
```

```
1 printf("masukkan 5 huruf kata : ");
```

```
1 printf("inputan tidak valid! coba lagi\n\n");
```

```
1 printf("inputan tidak valid! coba lagi\n\n");
```

```
1 printf("%c", kondisi[i]);
```

```
● ● ●  
1 printf("\n\n");
```

```
● ● ●  
1 printf("%c ", keyboard[i][j]);
```

```
● ● ●  
1 printf("TERIMA KASIH SUDAH BERMAIN\n");
```

```
● ● ●  
1 printf("%s\n", tebakkanUser);
```

```
● ● ●  
1 printf("SELAMAT ANDA MENANG || (%s)\n", jawaban);
```

```
1 printf("sisa tebakkan (%d)\n", 6 - jumlahTebakkan);
```

```
1 printf("YAHH ANDA KALAH || (%s)\n", jawaban);
```

```
1 printf("RESTAR THE GAME?\n");
2     printf("'y' atau 'Y' untuk MENGULANG game (tekan tombol apa saja selain 'y' atau 'Y' untuk KELUAR dari game) > ");
```

-Input

```
1 fgets(tebakkanUser, 7, stdin);
```

untuk menginput tebakkan.

```
1 scanf(" %c", &keinginanUserMengulang);
```

untuk user menginput mau game nya di ulang atau tidak.

3

Looping

20

```
● ● ●  
1   for (int i = 0; i < 3; i++) {  
2       if (i == 1) {  
3           printf("GAME TEBAK KATA\n");  
4       } else {  
5           for (int j = 0; j < 15; j++) {  
6               printf("=");  
7           }  
8           printf("\n");  
9       }  
10  }  
11 }
```

nested loop yang berguna untuk mencetak title/header

```
● ● ●  
1   for (int i = 0; i < 5; i++) {  
2       if (pengecekanJawaban[i] == tebakkanUser[i]) {  
3           kondisi[i] = 'V';  
4           pengecekanJawaban[i] = '-';  
5       }  
6   }  
7  
8   for (int i = 0; i < 5; i++) {  
9       if (kondisi[i] != 'V') {  
10          for (int j = 0; j < 5; j++) {  
11              if (tebakkanUser[i] == pengecekanJawaban[j]) {  
12                  kondisi[i] = '!';  
13                  pengecekanJawaban[j] = '-';  
14                  break;  
15              }  
16          }  
17      }  
18  }  
19 }
```

for loop dan nested loop yang berguna untuk mengecek kondisi status tebakan dari user.



```
1 for (int i = 0; i < 5; i++) {  
2     printf("%c", kondisi[i]);  
3     fflush(stdout);  
4     usleep(500000);  
5 }
```

Mencetak langsung element dari array kondisi dengan delay setiap pencetakan.



```
1 int status = 0;  
2  
3 for (int i = 0; i < 5; i++) {  
4     if (hurufDiTebakkan == jawaban[i]) {  
5         status = 1;  
6     }  
7 }
```

for loop yang mengecek apakah variabel *hurufDiTebakkan* ada di *jawaban*. Jika ada, status yang awalnya bernilai 0 berubah menjadi 1.

```
1 for (int i = 0; i < strlen(tebakanUser); i++) {  
2     char hurufDiTebakkan = tebakanUser[i];  
3  
4     for (int j = 0; j < 3; j++) {  
5         for (int k = 0; k < 10; k++) {  
6             if ((ngecekCharacterAdaDiString(hurufDiTebakkan, jawaban) == 0) && hurufDiTebakkan == keyboard[j][k]) {  
7                 keyboard[j][k] = '_';  
8             }  
9         }  
10    }  
11 }
```

nested for loop untuk mengecek apakah *hurufDiTebakkan* ada atau tidak di dalam jawaban. kebalikannya, jika *hurufDiTebakkan* tidak ada di *jawaban*, *keyboard* tersebut akan berubah menjadi ‘_’.

```
1 for (int i = 0; i < 3; i++) {  
2     for (int j = 0; j < 10; j++) {  
3         if (keyboard[i][j] != '\0') {  
4             printf("%c ", keyboard[i][j]);  
5         }  
6     }  
7     printf("\n");  
8 }
```

for loop untuk mengeprint seluruh element *keyboard*.

```
● ● ●  
1  for (int i = 0; i < 3; i++) {  
2      if (i == 1) {  
3          printf("TERIMA KASIH SUDAH BERMAIN\n");  
4      } else {  
5          for (int j = 0 ; j < 26; j++) {  
6              printf("=");  
7          }  
8          printf("\n");  
9      }  
10 }
```

nested for loop untuk mencetak bagian akhir/ penutup dari program.

```
● ● ●  
1  while(1) {  
2      printf("masukkan 5 huruf kata : ");  
3      fgets(tebakkanUser, 7, stdin);  
4      tebakkanUser[strcspn(tebakkanUser, "\n")] = 0;  
5      if (strlen(tebakkanUser) == 5) {  
6          return tebakkanUser;  
7      } else if (strlen(tebakkanUser) < 5) {  
8          printf("inputan tidak valid! coba lagi\n\n");  
9      }else {  
10         printf("inputan tidak valid! coba lagi\n\n");  
11         int c;  
12         while ((c = getchar()) != '\n' && c != EOF);  
13     }  
14 }
```

while loop untuk memastikan user selalu memasukkan input yang valid.

```
1 while(1) {
2     char* tebakkanUser = inputDanPemvalidasianInput();
3     printf("%s\n", tebakkanUser);
4
5     if (prosesPengecekan(jawaban, tebakkanUser) == 1) {
6
7         printf("SELAMAT ANDA MENANG || (%s)\n", jawaban);
8         free(tebakkanUser);
9         break;
10    }
11
12    printf("sisa tebakkan (%d)\n", 6 - jumlahTebakkan);
13
14    kondisiKeyboardDanPrintKeyboard(tebakkanUser, keyboard, jawaban);
15    jumlahTebakkan++;
16
17    if (jumlahTebakkan >= 7) {
18
19        printf("YAHH ANDA KALAH || (%s)\n", jawaban);
20        free(tebakkanUser);
21        break;
22    }
23
24    free(tebakkanUser);
25}
26 }
```

```
1 do {
2     char keyboard [3][10] = {
3         {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'},
4         {'g', 'f', 'd', 's', 'h', 'j', 'k', 'l', 'n', 'm'},
5         {'z', 'x', 'c', 'v', 'b', 'n', 'm', '\0', '\0', '\0'}
6     };
7     printHeader();
8     char* jawaban = PemilihanSesiJawaban();
9
10    int jumlahTebakkan = 0;
11
12    while(1) {
13
14        char* tebakkanUser = inputDanPemvalidasianInput();
15        printf("%s\n", tebakkanUser);
16
17        if (prosesPengecekan(jawaban, tebakkanUser) == 1) {
18
19            printf("SELAMAT ANDA MENANG || (%s)\n", jawaban);
20            free(tebakkanUser);
21            break;
22        }
23
24        printf("sisa tebakkan (%d)\n", 6 - jumlahTebakkan);
25
26        kondisiKeyboardDanPrintKeyboard(tebakkanUser, keyboard, jawaban);
27        jumlahTebakkan++;
28
29        if (jumlahTebakkan >= 7) {
30
31            printf("YAHUH ANDA KALAH || (%s)\n", jawaban);
32            free(tebakkanUser);
33            break;
34        }
35
36        free(tebakkanUser);
37    }
38
39    printf("RESTART THE GAME?\n");
40    printf("y atau Y untuk MENGULANG game (tekan tombol apa saja selain 'y' atau 'Y' untuk KELUAR dari game) > ");
41    scanf(" %c", &keinginanUserMengulang);
42    int c;
43    while ((c = getchar()) != '\n' && c != EOF);
44
45 } while (keinginanUserMengulang == 'y' || keinginanUserMengulang == 'Y');
```

4

Conditional

10

```
1  for (int i = 0; i < 3; i++) {  
2      if (i == 1) {  
3          printf("GAME TEBAK KATA\n");  
4      } else {  
5          for (int j = 0; j < 15; j++) {  
6              printf("=");  
7          }  
8          printf("\n");  
9      }  
10 }  
11 }
```

```
1  if (strlen(tebakkanUser) == 5) {  
2      return tebakkanUser;  
3  } else if (strlen(tebakkanUser) < 5) {  
4      printf("inputan tidak valid! coba lagi\n\n");  
5  }else {  
6      printf("inputan tidak valid! coba lagi\n\n");  
7      int c;  
8      while ((c = getchar()) != '\n' && c != EOF);  
9  }
```

```
● ● ●  
1 if (pengecekanJawaban[i] == tebakkanUser[i]) {  
2  
3     kondisi[i] = 'V';  
4     pengecekanJawaban[i] = '-';  
5 }
```

```
● ● ●  
1 if (kondisi[i] != 'V') {  
2  
3     for (int j = 0; j < 5; j++) {  
4         if (tebakkanUser[i] == pengecekanJawaban[j]) {  
5             kondisi[i] = '!';  
6             pengecekanJawaban[j] = '-';  
7             break;  
8         }  
9     }  
10 }  
11 }  
12 }
```

```
● ● ●  
1 if (strcmp(jawaban, tebakkanUser) == 0) {  
2     return 1;  
3 } else {  
4     return 0;  
5 }
```

```
● ● ●  
1     if (hurufDiTebakkan == jawaban[i]) {  
2  
3         status = 1;  
4     }  
5
```

```
● ● ●  
1     if ((ngecekCharacterAdaDiString(hurufDiTebakkan, jawaban) == 0) && hurufDiTebakkan == keyboard[j][k]) {  
2  
3         keyboard[j][k] = '_';  
4     }  
5
```

```
● ● ●  
1     if (keyboard[i][j] != '\0') {  
2  
3         printf("%c ", keyboard[i][j]);  
4     }
```

```
● ● ●  
1     if (i == 1) {  
2  
3         printf("TERIMA KASIH SUDAH BERMAIN\n");  
4     } else {  
5         for (int j = 0 ; j < 26; j++) {  
6             printf("=");  
7         }  
8         printf("\n");  
9     }  
10    }  
11}
```

			<pre> ● ● ● 1 if (prosesPengecekan(jawaban, tebakkanUser) == 1) { 2 3 printf("SELAMAT ANDA MENANG (%s)\n", jawaban); 4 free(tebakkanUser); 5 break; 6 } </pre>
			<pre> ● ● ● 1 if (jumlahTebakkan >= 7) { 2 3 printf("YAHH ANDA KALAH (%s)\n", jawaban); 4 free(tebakkanUser); 5 break; 6 } </pre>
6	Array	25	<pre> ● ● ● 1 char* listJawaban[] = { 2 "absen", "asbak", "antik", "badai", "badut", "bagus", 3 "bahas", "bahau", "bahwa", "balok", "bantu", "bapak", 4 "baris", "batas", "batik", "beras", "biasa", "bidan", 5 "bisik", "butir", "canda", "catat", "celah", "cemas", 6 "citra", "damai", "datar", "debut", "dekat", "denda", 7 "darah", "dunia", "enzim", "etika", "fakta", "fokus", 8 "gagal", "ganda", "garam", "gelar", "gempa", "goyah", 9 "hadir", "harga", "hasil", "hujan", "iklan", "indah", 10 "jarak", "jenuh", "kabar", "kadal", "kawat", "kenal", 11 "kolam", "kotak", "lihat", "lidah", "macan", "mandi", 12 "maret", "minta", "mobil", "nanti", "nyata", "ombak", 13 "pahat", "pasti", "petir", "pinus", "pulau", "ramah", 14 "rapat", "rehat", "rumah", "sahur", "senam", "siang", 15 "susun", "tanda", "tebal", "tipis" 16 }; </pre> <pre> ● ● ● 1 char kondisi[] = {'.', '.', '.', '.', '.', '.'}; </pre>



```
1 char pengecekanJawaban[6];
```



```
1 char Keyboard [3][10] = {  
2     {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'},  
3     {'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', '\0'},  
4     {'z', 'x', 'c', 'v', 'b', 'n', 'm', '\0', '\0', '\0'}};
```

Total : 70

NB : +10 poin kerapian, +20 poin kelengkapan laporan