

LAPORAN TUGAS BESAR
IF2111 Algoritma dan Struktur Data

BNMO


Dipersiapkan oleh:

Kelompok 13

18219048 Dwiky Hared Darmawan
18221162 Ceavin Rufus De Prayer Purba
18221046 Vincent Winarta
18221084 Rei Arriel Clyfton
18221118 Ardhan Nur Urfan
18221128 Fadhlhan Nazhif Azizy

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB1-13		40
		Revisi	-	11 November 2022

Daftar Isi

1 Ringkasan	3
2 Penjelasan Tambahan Spesifikasi Tugas	4
2.1 Bonus Game (Tictactoe)	4
2.2 Bonus Game (Tower of Hanoi)	5
3 Struktur Data (ADT)	5
3.1 ADT Array	5
3.2 ADT Arraydin	5
3.3 ADT Word	5
3.4 ADT Mesin Karakter dan Mesin Kata	5
3.5 ADT String	6
3.6 ADT Queue	6
3.7 ADT Stack	6
3.8 ADT Matrix	6
3.9 ADT Masakan	6
3.10 ADT Order (Variasi ADT Queue)	7
4 Program Utama	7
5 Algoritma-Algoritma Menarik	8
5.1 Akuisisi kata pada urutan tertentu dalam command	8
5.2 Looping pada matrix tanpa nested loop	9
6 Data Test	9
6.1 Start	9
6.2 List Game	10
6.3 Create Game	11
6.4 Queue Game	12
6.5 Play Game	12
6.6 Skip Game	13
6.7 Delete Game	13
6.8 Help	14
6.9 Save	15
6.10 Quit	15
6.11 Load	16
6.12 Command Lain	17

6.13 Diner Dash	17
6.14 Tower of Hanoi	21
6.15 Tictactoe	24
6.16 RNG	26
7 Test Script	27
8 Pembagian Kerja dalam Kelompok	31
9 Lampiran	33
9.1 Deskripsi Tugas Besar 2	33
9.2 Notulen Rapat	34
9.3 Log Activity Anggota Kelompok	35
9.4 MoM Asistensi Tugas Besar	36

1 Ringkasan

Dengan niat membantu Indra dan Doni dalam memperbaiki sistem robot *video game console* milik mereka, yaitu BNMO. Kami sebagai programmer handal yang dicari oleh Indra dan Doni berinisiatif untuk memprogram ulang BNMO tersebut dengan cara membuat sebuah permainan berbasis CLI (*Command Line Interface*).

Dalam laporan ini berisi penjelasan mengenai program yang telah dibuat, antara lain spesifikasi program, yang mencakup tipe data bentukan (ADT) yang digunakan pada program ini, algoritma-algoritma khusus yang terdapat pada laporan, data tes program, dan *test script* saat kami melakukan testing. Pada program ini tentunya terdapat program utama atau *main program* yang digunakan sebagai wadah sinkronisasi antar program *command* sehingga program BNMO dapat dijalankan secara utuh dan selaras. Hal - hal mengenai program utama juga akan dijelaskan secara lebih lanjut pada laporan ini. Dalam laporan juga terlampir pembagian kerja dalam kelompok, notulen rapat, deskripsi tugas besar, dan *log activity* anggota kelompok.

Program ini dibuat menggunakan bahasa pemrograman C dan dengan memanfaatkan ADT yang sudah dipelajari pada mata kuliah IF2111 Algoritma dan Struktur Data STI, seperti ADT Array dinamis, ADT Mesin kata, ADT mesin karakter, dan ADT lainnya. BNMO menggunakan tampilan command line interface dan untuk menggunakannya hanya perlu memasukkan command dari pemain. Beberapa kegunaan dari robot console ini antara lain memainkan *game*, menambahkan *game*, menghapus *game*, dan mengurutkan *game* yang akan dimainkan. Semua perubahan yang kita lakukan terhadap program *game* pada saat bermain seperti menambahkan dan juga menghapus *game* dapat disimpan pada sebuah *file* dengan *extension* '.txt'. Dalam program ini, terdapat dua *game* yang secara *default* dapat dimainkan oleh pengguna, yaitu RNG dan Diner Dash. Namun, pada robot console yang kami buat, terdapat *game* tambahan yang juga dapat dimainkan oleh pengguna, yaitu Tictactoe dan Tower of Hanoi.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Bonus Game (Tictactoe)

Bonus *game* yang dibuat oleh kelompok kami adalah permainan Tictactoe. Di dalam permainan ini terdapat beberapa fungsi dan prosedur. Fungsi *isWin* adalah fungsi untuk mengecek apakah permainan sudah dimenangkan oleh salah satu *player* atau belum. Dalam Tictactoe, untuk menentukan *player* menang atau tidak, terdapat beberapa kondisi yang di dalam program kami, kami dekomposisi menjadi masing-masing fungsi, yaitu *isHorizontalWin*, *isVerticalWin*, dan *isDiagonalWin*.

Kemudian, ada prosedur board yang merupakan prosedur yang dibuat untuk mencetak papan permainan yang terdiri dari 9 kotak dan diberi angka 1 sampai dengan 9. Terakhir, ada prosedur tictactoe yang merupakan prosedur untuk memulai permainan. Pemenang adalah *player* yang mendapatkan 3 mark berurut secara horizontal, vertikal, atau diagonal. Jika papan penuh, tetapi belum ada pemain yang menang, kedua *player* dianggap seri.

2.2 Bonus Game (Tower of Hanoi)

Bonus *game* kedua yang dibuat oleh kelompok kami adalah permainan Tower of Hanoi. Di dalam permainan ini ada 2 prosedur untuk mencetak *state* dari setiap *tower*, yaitu prosedur DisplayState dan PrintDisk. Selain itu, terdapat prosedur towerofhanoi untuk memulai permainan. Permainan akan berakhir ketika semua disk pada tower 1 sudah dipindahkan ke tower 3 dengan ketentuan ukuran disk terurut membesar dimulai dari elemen top.

3 Struktur Data (ADT)

3.1 ADT Array

ADT array merupakan tipe data bentukan yang terdiri dari array TI (array berjenis statik) sebagai tempat menyimpan elemen dan Neff yang merupakan banyaknya elemen efektif pada array tersebut. ADT disimpan pada *file array.c*. ADT ini digunakan dalam *game* Diner Dash untuk menyimpan daftar masakan yang sedang dimasak dan yang sudah siap disajikan.

3.2 ADT Arraydin

ADT arraydin merupakan tipe bentukan yang terdiri dari Capacity untuk menyimpan kapasitas maksimum, Neff menyimpan banyak elemen efektif dan suatu buffer A. A memiliki tipe pointer to Word. ADT ini digunakan karena pada program LOAD, pembacaan file digunakan dengan memanfaatkan ADT mesinkata. Sehingga, tidak perlu dilakukan banyak casting dari ADT ke ADT lain ataupun ke bentuk selain Word. ADT ini digunakan pada file console.c dan main.c .

3.3 ADT Word

ADT Word digunakan untuk mengolah sebuah kata. sebuah Word terdiri dari sebuah string statik Tabword dan panjang efektifnya. Fungsi-fungsi pada ADT Word digunakan untuk casting dari bentuk word ke bentuk lain ataupun melakukan copyWord. ADT word dipisahkan dari mesinkata karena sebagian program menyimpan data dalam bentuk Word sehingga akan banyak dilakukan pengolahan kepada Word.

3.4 ADT Mesin Karakter dan Mesin Kata

ADT mesin karakter adalah tipe data bentukan yang dapat membaca sebuah pita karakter panjang. Mekanisme kerjanya dengan membaca karakter satu per satu yang disimpan dalam sebuah variabel cc (current character) hingga pita karakter habis yang ditandai dengan keadaan end of process (eop).

ADT mesin karakter akan digunakan di dalam ADT mesin kata. ADT mesin kata menjalankan kata dalam bentuk Word secara satu persatu dan dapat membawa kata yang disimpan dalam suatu variabel currentWord dan diakhiri keadaan endofWord keadaan pembacaan pita telah selesai.

3.5 ADT String

ADT string sebenarnya merupakan pointer to first character. Kami juga membuat beberapa primitif dari ADT string ini untuk memudahkan kami dalam bekerja dengan string (karena tidak diperbolehkan memakai library <string.h>).

3.6 ADT Queue

ADT queue merupakan ADT yang merepresentasikan sebuah antrian. Dalam antrian tersebut berlaku FIFO yakni *first in first out*. Dalam hal ini apabila terjadi penambahan suatu elemen maka akan dilakukan penambahan pada urutan paling belakang, sebaliknya apabila terjadi penghapusan suatu elemen, elemen paling depan yang akan diproses. Dalam kasus kali ini ADT queue digunakan pada sistem antrian game yang akan dimainkan. Pengguna akan menambahkan antrian game yang akan dimainkan, kemudian akan memainkan sesuai urutan pertama dalam antrian.

3.7 ADT Stack

ADT stack merupakan ADT yang merepresentasikan sebuah tumpukan. Dalam tumpukan berlaku LIFO (*last in last out*). Dalam hal ini apabila terjadi penambahan suatu elemen akan dilakukan dengan cara menambahkan elemen pada urutan paling atas, begitu juga dengan penghapusan suatu elemen, penghapusan akan dilakukan pada urutan paling atas. ADT stack ini akan digunakan pada *game* Tower of Hanoi yang merupakan permainan yang sangat berkaitan dengan tumpukan ini.

Untuk mengakomodasi kebutuhan pada game Tower of Hanoi, ADT stack yang digunakan di sini kami modifikasi pada bagian pop dan push. Pada kedua prosedur ditambahkan parameter baru, yaitu *succeed*, untuk mengetahui apakah pop atau push bekerja dengan baik. Pada bagian pop, initial state dari stack boleh kosong, tetapi jika dilakukan pop pada stack yang kosong prosedur hanya akan melakukan pengubahan pada nilai *succeed* menjadi false. Pada bagian push, terjadi sedikit perubahan pada aturan penambahan elemen, yaitu elemen yang ingin ditambahkan dalam suatu stack harus lebih kecil dari elemen TOP pada stack tersebut. Jika memenuhi syarat tersebut, prosedur akan mengubah nilai *succeed* menjadi false.

3.8 ADT Matrix

ADT matriks merupakan suatu kumpulan elemen yang direpresentasikan dalam bentuk baris dan kolom. Dalam kasus kali ini ADT matrix digunakan untuk mengimplementasikan sebuah game Tictactoe. Dalam hal ini matrix yang kami gunakan berukuran 3x3. ADT ini dipilih karena Tictactoe merupakan game yang merepresentasikan matrix yang nantinya pemain akan menyusun elemen yang sama dalam satu baris, satu kolom, atau satu diagonal. Oleh karena itu, ADT Matrix dinilai efisien untuk digunakan dalam mengimplementasikan game Tictactoe.

3.9 ADT Masakan

ADT masakan merupakan tipe data bentukan pasangan/binding key dengan multiple value. Key dari ADT masakan ini adalah nomor masakan, dan untuk valuenya adalah durasi, ketahanan, dan harga.

3.10 ADT Order (Variasi ADT Queue)

ADT order adalah tipe data bentukan yang merupakan variasi dari ADT queue. Pada ADT order ini, tipe elemen yang ada pada buffer penampung elemen adalah tipe data masakan (penjelasan ADT masakan pada 3.6). Perbedaan lainnya adalah dalam ADT order ini terdapat primitif untuk mengecek apakah suatu masakan terdapat di dalam order, mengetahui urutan suatu masakan dalam suatu order, dan untuk mengetahui masakan pada urutan tertentu. Primitif-primitif ini akan digunakan dalam *game* Diner Dash, di mana untuk memasak sebuah masakan dalam sebuah order perlu mengambil elemen di tengah order.

4 Program Utama

Program utama berfungsi sebagai driver program lain dengan memasukkan fungsi yang telah dibuat yaitu, `string.h`, `arraydin.h`, `queue.h`, `word.h`, `mesinkata.h`, dan `mesin karakter.h`. Pada program utama juga terdapat fungsi main menu yang menampilkan opsi START dan LOAD. Ketika program dijalankan, program akan meminta pemain untuk memasukkan START atau LOAD. Ketika pemain memasukkan START atau LOAD yang valid maka program akan dimulai dan menampilkan perintah-perintah yang dapat dimasukkan oleh pemain. Jika perintah tidak valid maka akan menampilkan pesan masukkan tidak valid dan pemain diminta memasukkan perintah lagi.

Pada control loop pemain akan diminta untuk memasukkan perintah yang valid. SAVE untuk menyimpan file berisi state game dari pemain.

- CREATEGAME adalah command untuk menambahkan game ke dalam list game.
- LISTGAME adalah command untuk menampilkan list game yang tersedia di program.
- QUEUEGAME adalah command untuk menampilkan antrian game dan juga menampilkan list game lalu pemain dapat menambahkan game dengan memasukkan angka game untuk dimasukkan ke dalam antrian game yang ingin dimainkan.
- PLAYGAME adalah command untuk menjalankan game pada urutan pertama sesuai dengan antrian game. Game yang dapat dimainkan hanya RNG, Diner Dash, Tictactoe, dan Tower of Hanoi. Selain game-game tersebut, program akan memberikan pesan game tidak tersedia.
- SKIPGAME adalah command untuk melewati game sesuai dengan masukan pemain.
- DELETGAME adalah command untuk menghapus game pada list game.
- HELP adalah command untuk menampilkan fungsi dari command-command yang ada.
- QUIT adalah command untuk keluar dari program dan menghapus state game pemain yang tidak disimpan
-

Setelah pengguna selesai menggunakan program tentu pengguna akan melakukan command QUIT untuk keluar dari program. Pada saat menginput command QUIT, maka akan ada dua pilihan yaitu untuk menyimpan file yang sudah diproses saat menggunakan program atau tidak menyimpan file.

5 Algoritma-Algoritma Menarik

5.1 Akuisisi kata pada urutan tertentu dalam command

```
void akuisisiCommandWord(Word *w, Word command, int kataKe)
{
    int i = 0, counter = 0, length = 0;
    boolean stop;

    while (counter != kataKe - 1 && i < command.Length)
    {
        stop = false;
        if (command.TabWord[i] == ' ')
        {
            counter++;
            while (i < command.Length && !stop)
            {
                i++;
                if (command.TabWord[i] != ' ')
                {
                    stop = true;
                }
            }
        }
        else
        {
            i++;
        }

        if (i == command.Length)
        {
            counter++;
        }
    }

    stop = false;
    while (!stop && i < command.Length)
    {
        if (command.TabWord[i] == ' ')
        {
            stop = true;
        }
        else
        {
            w->TabWord[length] = command.TabWord[i];
            i++;
            length++;
        }
    }
    w->Length = length;
}
```

Gambar 5.1.1

Algoritma ini digunakan untuk parsing input dari user. Algoritma ini terbagi menjadi 2 bagian. Pada bagian pertama, akan dilakukan iterasi sampai kata yang telah dilewati sebanyak kataKe - 1. Pada algoritma ini juga akan mengabaikan spasi, sehingga program hanya akan mengakuisisi *non-whitespace character*. Pada bagian kedua, dilakukan iterasi kata pada urutan kataKe untuk menyalin kata tersebut. Iterasi akan berhenti jika ditemukan spasi atau sudah mencapai akhir dari

word. Pada setiap iterasi akan memasukan setiap karakter pada command ke dalam word yang menjadi parameter input/output dari prosedur ini. Algoritma ini menarik karena penerimaan dan parsing input biasanya dilakukan dengan menggunakan scanf. Pada kasus ini kami menerima input menggunakan stdin, sehingga perlu dibuat prosedur khusus untuk parsing input yang diterima. Alasan kedua mengapa algoritma ini menarik adalah karena algoritma ini sekaligus mengabaikan *whitespace* yang terdapat pada input.

5.2 *Looping pada matrix tanpa nested loop*



Gambar 5.2.1

Gambar 5.2.1 merupakan salah satu pemakaian algoritma *looping* pada matriks tanpa *nested loop* dalam program yang telah kami buat. Algoritma ini menarik karena biasanya iterasi pada matriks dilakukan dengan menggunakan *nested loop*. Selain itu, karena tidak menggunakan *nested loop*, menurut kami algoritma ini lebih clean dibanding dengan menggunakan *nested loop*. Algoritma ini digunakan saat melakukan assignment atau pengecekan pada seluruh elemen matriks.

6 Data Test

6.1 *Start*

Fitur yang dites pada saat program di Run dan dimasukkan command “START”, program akan mulai menjalankan mesin karakter untuk array dinamik arrGame. Selanjutnya, program akan melanjutkan dan menampilkan ke main menu.


```

ENTER COMMAND: LIST GAME
Berikut adalah daftar game yang tersedia
1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. Tictactoe
7. TOWER OF HANOI

```

Gambar 6.2.1

6.3 Create Game

Perintah “CREATE GAME” digunakan untuk menambahkan nama game pada `arrGame` yang akan dibaca pada *function* LISTGAME. Pada testing kali ini, kami menambahkan game bernama Bomb. Dengan begitu saat ini terdapat 8 game yang berada di LIST GAME.

```

===== MAIN MENU =====
1. SAVE <nama_file>
2. CREATE GAME
3. LIST GAME
4. DELETE GAME
5. QUEUE GAME
6. PLAY GAME
7. SKIP GAME <n>
8. QUIT
9. HELP

ENTER COMMAND: CREATE GAME
Masukkan nama game yang akan ditambahkan: Bomb
Game berhasil ditambahkan

```

Gambar 6.3.1

Kemudian kami melakukan testing lagi dengan menambahkan ulang game bernama Bomb. Maka, program gagal menambahkan karena game Bomb sudah ada dalam list game.

```

===== MAIN MENU =====
1. SAVE <nama_file>
2. CREATE GAME
3. LIST GAME
4. DELETE GAME
5. QUEUE GAME
6. PLAY GAME
7. SKIP GAME <n>
8. QUIT
9. HELP

ENTER COMMAND: CREATE GAME
Masukkan nama game yang akan ditambahkan: Bomb
Game sudah tersedia.

```

Gambar 6.3.2

6.4 Queue Game

Perintah “QUEUE GAME” akan menampilkan data pada arrQueue (game yang sudah di queue sebelumnya) dan arrGame(list game yang tersedia). Lalu, akan diminta masukan angka dari urutan list game yang akan dimasukan ke dalam arrQueue. Input bisa tidak valid jika nilai input lebih besar dari jumlah game yang tersedia. Pada testing kali ini, kita ingin memasukan game RNG dengan memasukan input 1. Dikarenakan 1 adalah valid, maka terdapat keterangan bahwa game telah berhasil dimasukan ke dalam antrian.

```
ENTER COMMAND: QUEUE GAME
Berikut adalah daftar antrian game-mu

===== Daftar Kosong =====

Berikut adalah daftar game yang tersedia
1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. Tictactoe
7. TOWER OF HANOI
8. Bomb

Nomor Game yang mau ditambahkan ke antrian: 1
Game berhasil ditambahkan kedalam daftar antrian.
```

Gambar 6.4.1

Dikarenakan game RNG sudah berada pada queue, sehingga jika kita memasukan perintah QUEUE GAME kembali, terdapat game RNG yang ditampilkan pada urutan.

```
ENTER COMMAND: QUEUE GAME
Berikut adalah daftar antrian game-mu
1. RNG

Berikut adalah daftar game yang tersedia
1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. Tictactoe
7. TOWER OF HANOI
8. Bomb

Nomor Game yang mau ditambahkan ke antrian: 1
Game berhasil ditambahkan kedalam daftar antrian.
```

Gambar 6.4.2

6.5 Play Game

Perintah “PLAY GAME” akan menjalankan game sesuai dengan urutan Queue game yang disimpan pada arrQueue. Dalam kasus ini, game RNG ada pada Queue urutan pertama, sehingga saat perintah tersebut dijalankan, user akan disajikan game RNG.

```
ENTER COMMAND: PLAY GAME
Berikut adalah daftar Game-mu
1. RNG

Loading RNG ...

=====
```

Gambar 6.5.1

6.6 Skip Game

Perintah “SKIP GAME” yang akan disertai dengan angka, dimana angka tersebut menjadi parameter game yang sudah di queue, yang akan dilewati urutannya dan langsung di play. Nilai input akan valid jika nilai tersebut maksimal sebesar jumlah game yang di queue-1. Pada kasus test ini, user telah menambahkan game “DINOSAUR IN EARTH”, “DINER DASH”, & “RNG”. Setelah pengguna memasukan perintah “SKIP GAME 2”, maka akan 2 game secara urutan yang akan dilewati. Dengan begitu, BNMO akan menjalankan game RNG.

```
ENTER COMMAND: SKIP GAME 2
Berikut adalah daftar Game-mu
1. DINOSAUR IN EARTH
2. Diner DASH
3. RNG

Loading RNG ...
```

Gambar 6.6.1

6.7 Delete Game

Perintah “DELETE GAME” digunakan untuk menghapus game yang ada pada LIST GAME yang disimpan pada arrGame. Setelah memasukan perintah tersebut, pengguna akan diminta memasukan urutan dari LIST GAME yang akan dihapus. Nilai masukan akan valid jika pengguna tidak menghapus game *default* (urutan 1-5) dan nilai maksimal input adalah jumlah dari game yang ada pada LIST GAME. Jumlah game pada LIST GAME diketahui dari Neff pada arrGame.

Pada kasus di bawah (Gambar 6.7.1), game nomor 6 (Tictactoe) akan dihapus. Dikarenakan bukan game default, maka game tersebut berhasil dihapus.

```

ENTER COMMAND: DELETE GAME
Berikut adalah daftar game yang tersedia

1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. Tictactoe
7. TOWER OF HANOI
8. Bomb

Masukan game yang akan dihapus: 6

Game berhasil dihapus

```

Gambar 6.7.1

Pada kasus ini, pengguna mencoba untuk menghapus game ke-3 (DINOSAUR IN EARTH) yang merupakan game *default* yang tidak bisa dihapus.

```

ENTER COMMAND: DELETE GAME
Berikut adalah daftar game yang tersedia

1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. TOWER OF HANOI
7. Bomb

Masukan game yang akan dihapus: 3

Game gagal dihapus

```

Gambar 6.7.2

6.8 Help

Perintah “HELP” akan menampilkan kegunaan setiap command-command yang tersedia pada program utama.

```

ENTER COMMAND: HELP

===== HALAMAN HELP =====
=====

GUNAKAN COMMAND BERIKUT UNTUK BERINTERAKSI DENGAN PROGRAM INI
1. START      : Memulai program
2. LOAD <nama_file> : Membuka savefile
3. SAVE <nama_file> : Menyimpan
4. CREATE GAME : Menambahkan game
5. LIST GAME   : Menampilkan daftar game
6. DELETE GAME : Menghapus daftar game
7. QUEUE GAME  : Menambahkan game ke daftar antrian yang akan dimainkan
8. PLAY GAME   : Memainkan game pada daftar antrian paling atas
9. SKIPGAME <n> : Memainkan game dengan mendahului beberapa game di atasnya
10. QUIT       : Keluar dari program
11. HELP       : Panduan penggunaan

```

Gambar 6.8.1

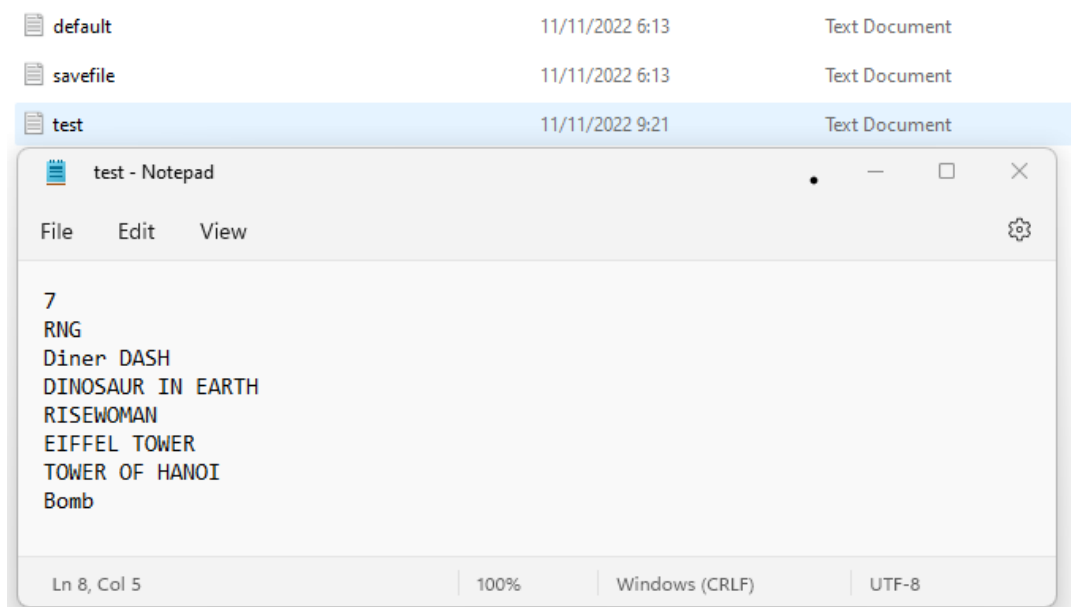
6.9 Save

Perintah “SAVE” digunakan untuk menyimpan data LIST GAME terakhir pada program ke dalam folder “data” dengan format file .txt. Untuk percobaan kali ini, menggunakan filename test.txt.

```
ENTER COMMAND: SAVE test.txt  
Save file berhasil disimpan..
```

Gambar 6.9.1

Pada testing kali ini, game Tictactoe telah dihapus menggunakan *function* “DELETE GAME”. Dengan begitu, hanya terdapat 7 game yang terdiri dari 5 game *default*, 2 game tambahan, dan terdapat game “Bomb” hasil dari *function* CREATE GAME.



Gambar 6.9.2

6.10 Quit

Perintah “QUIT” akan membuat pengguna keluar dari program BNMO. Dengan begitu, arrQueue juga akan terhapus.

```

1. SAVE <nama_file>
2. CREATE GAME
3. LIST GAME
4. DELETE GAME
5. QUEUE GAME
6. PLAY GAME
7. SKIPGAME <n>
8. QUIT
9. HELP

ENTER COMMAND: QUIT

```

Gambar 6.10.1

6.11 Load

Perintah “LOAD” dapat digunakan jika LIST GAME yang akan digunakan bukanlah pada *default*. Sehingga, pengguna akan load file txt yang diinginkan. Pada testing kali ini, kita menggunakan file txt yang telah disimpan menggunakan *function* “SAVE” yang dilakukan sebelumnya dengan nama file test.txt. (Data test 6.9).

```

===== MAIN MENU =====
1. START
2. LOAD <nama_file>

ENTER COMMAND: LOAD test.txt
Save file berhasil dibaca. BNMO berhasil dijalankan.

```

Gambar 6.11.1

Dengan begitu, LIST GAME yang ditampilkan akan sesuai dengan file test.txt, dimana pada kesempatan itu, pengguna telah menambahkan game bernama “Bomb” dan menghilangkan (DELETE GAME) “Tictactoe”.

```

ENTER COMMAND: LIST GAME
Berikut adalah daftar game yang tersedia
1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. TOWER OF HANOI
7. Bomb

```

Gambar 6.11.2

Jika nama file yang dimasukan tidak ada pada folder “data”, seperti pada test kali ini tidak ada file yang bernama coba.txt, maka input akan di-*reject* dan pembacaan file akan gagal.


```

===== MAIN MENU =====
1. START
2. LOAD <nama_file>

ENTER COMMAND: LOAD coba.txt
Gagal membaca file.

```

Gambar 6.11.3

6.12 Command Lain

Jika pengguna memasukan lain yang tidak terdapat pada *command* yang tersedia, maka command tersebut tidak dikenali dan program akan diminta untuk memasukan *command* yang valid. Seperti pada testing diatas, *command* “MAIN GAME” tidak dikenali sehingga pengguna diminta memasukan kembali *command* yang dimaksud.

```

ENTER COMMAND: MAIN GAME
Command tidak dikenali, silahkan masukkan command yang valid.

```

Gambar 6.12.1

6.13 Diner Dash

Fitur yang dites dalam *data test* ini merupakan fitur game dari diner dash secara keseluruhan. seperti perintah “COOK”, “SERVE”, dan “SKIP”. Selain itu, juga akan dilakukan pengecekan pada kondisi akhir permainan.

Pada testing ini, dilakukan testing command “COOK”. Jika kita mengetik COOK, diikuti dengan parameter makanan yang ingin dimasak, makanan tersebut akan dimasak dan akan masuk ke dalam tabel makanan yang sedang dimasak.

```

=====
SALDO: 0

Daftar Pesanan
Makanan      | Durasi memasak      | Ketahanan      | Harga
-----
M0           | 2                   | 4              | 10589
M1           | 3                   | 2              | 13857
M2           | 3                   | 3              | 11273

Daftar Makanan yang sedang dimasak
Makanan      | Sisa durasi memasak
-----
|

Daftar Makanan yang dapat disajikan
Makanan      | Sisa ketahanan makanan
-----
|

MASUKKAN COMMAND: COOK M0
Berhasil memasak M0

```

Gambar 6.13.1

Namun, ketika memberikan command COOK dan parameter makanan yang ingin dimasak tidak ada di daftar pesanan seperti COOK M4, input akan direject oleh program karena dianggap tidak

valid. Sebenarnya di program ini masih banyak handling input tidak valid lainnya, tetapi untuk mempersingkat, yang dicantumkan pada laporan hanya beberapa kasus saja (berlaku untuk semua data test).

```
=====
SALDO: 0

Daftar Pesanan
Makanan | Durasi memasak | Ketahanan | Harga
-----
M0      | 2              | 4         | 10589
M1      | 3              | 2         | 13857
M2      | 3              | 3         | 11273
M3      | 4              | 5         | 10663

Daftar Makanan yang sedang dimasak
Makanan | Sisa durasi memasak
-----
M0      | 2

Daftar Makanan yang dapat disajikan
Makanan | Sisa ketahanan makanan
-----
|

MASUKKAN COMMAND: COOK M4
M4 tidak ada di pesanan
```

Gambar 6.13.2

Kemudian, akan dilakukan testing ketika durasi memasak sebuah masakan telah selesai. Seperti pada Gambar 6.13.3, ketika M0 selesai dimasak, program akan memberitahu user dan M0 otomatis masuk ke daftar makanan yang dapat disajikan.

```
=====
SALDO: 0

Daftar Pesanan
Makanan | Durasi memasak | Ketahanan | Harga
-----
M0      | 2              | 4         | 10589
M1      | 3              | 2         | 13857
M2      | 3              | 3         | 11273
M3      | 4              | 5         | 10663
M4      | 1              | 1         | 21790

Daftar Makanan yang sedang dimasak
Makanan | Sisa durasi memasak
-----
M0      | 1
M2      | 3

Daftar Makanan yang dapat disajikan
Makanan | Sisa ketahanan makanan
-----
|

MASUKKAN COMMAND: COOK M1
Berhasil memasak M1
Makanan M0 telah selesai dimasak
```

Gambar 6.13.3

Kemudian karena sudah terdapat makanan pada daftar makanan yang dapat disajikan, akan dilakukan testing command “SERVE”. Jika kita mengetik SERVE, diikuti dengan parameter makanan yang ingin disajikan, makanan tersebut akan disajikan dan dihapus dari daftar makanan yang dapat disajikan, dengan syarat makanan tersebut berada di urutan pertama daftar pesanan.

```
=====
SALDO: 0

Daftar Pesanan
Makanan      | Durasi memasak      | Ketahanan      | Harga
-----
M0            | 2                    | 4               | 10589
M1            | 3                    | 2               | 13857
M2            | 3                    | 3               | 11273
M3            | 4                    | 5               | 10663
M4            | 1                    | 1               | 21790
M5            | 4                    | 3               | 30262

Daftar Makanan yang sedang dimasak
Makanan      | Sisa durasi memasak
-----
M1            | 3
M2            | 2

Daftar Makanan yang dapat disajikan
Makanan      | Sisa ketahanan makanan
-----
M0            | 4

MASUKKAN COMMAND: SERVE M0

Berhasil mengantarkan M0
```

Gambar 6.13.4

Sambil menunggu makanan yang sedang dimasak, akan dilakukan testing pada command “SKIP”, yaitu program akan melanjutkan putaran meskipun tidak ada perubahan state.

```

=====
SALDO: 10589

Daftar Pesanan
Makanan      | Durasi memasak      | Ketahanan      | Harga
-----
M1            | 3                   | 2              | 13857
M2            | 3                   | 3              | 11273
M3            | 4                   | 5              | 10663
M4            | 1                   | 1              | 21790
M5            | 4                   | 3              | 30262
M6            | 5                   | 2              | 29650

Daftar Makanan yang sedang dimasak
Makanan      | Sisa durasi memasak
-----
M1            | 2
M2            | 1

Daftar Makanan yang dapat disajikan
Makanan      | Sisa ketahanan makanan
-----
|

MASUKKAN COMMAND: SKIP
Makanan M2 telah selesai dimasak

```

Gambar 6.13.5

Setelah dilakukan SKIP dan makanan M2 selesai dimasak, akan dilakukan testing apakah M2 dapat disajikan (seharusnya tidak bisa karena M2 tidak terletak pada urutan pertama daftar pesanan).

```

=====
SALDO: 10589

Daftar Pesanan
Makanan      | Durasi memasak      | Ketahanan      | Harga
-----
M1            | 3                   | 2              | 13857
M2            | 3                   | 3              | 11273
M3            | 4                   | 5              | 10663
M4            | 1                   | 1              | 21790
M5            | 4                   | 3              | 30262
M6            | 5                   | 2              | 29650
M7            | 4                   | 2              | 26077

Daftar Makanan yang sedang dimasak
Makanan      | Sisa durasi memasak
-----
M1            | 1

Daftar Makanan yang dapat disajikan
Makanan      | Sisa ketahanan makanan
-----
M2            | 3

MASUKKAN COMMAND: SERVE M2
M2 belum dapat disajikan karena M1 belum selesai

```

Gambar 6.13.6

Seperti yang diharapkan terjadi, M2 tidak bisa disajikan karena bukan berada di urutan pertama daftar pesanan.

Kemudian, akan dilakukan testing akhir dari permainan. Permainan akan berakhir jika banyak antrian melebihi 7 atau pesanan yang berhasil disajikan mencapai 15. Pada Gambar 6.13.6, antrian pesanan sudah mencapai 7, maka jika diberikan command “SKIP” seharusnya state akan berubah dan permainan akan berakhir (karena pesanan bertambah 1 dan daftar pesanan melebihi 7).

```
MASUKKAN COMMAND: SKIP
Makanan M1 telah selesai dimasak
Kamu kalah karena antrian telah melebihi 7 pelanggan
Skor: 10589
```

Gambar 6.13.7

6.14 Tower of Hanoi

Pada data test ini akan dilakukan pengetesan pada seluruh fitur permainan Tower of Hanoi.

Pada awal permainan, akan diminta input berapa banyak disk yang ingin ada pada tower 1, dan nilainya hanya berkisar 3-8 saja. Jika input bukan berada pada rentang tersebut, program akan meminta input kembali sampai input valid. Jika sudah valid, program akan menampilkan 3 tower dengan jumlah disk sesuai input.

```
Banyak disk (3-8): 2
Banyak disk (3-8): 3

Moves: 0
Minimum moves to solve: 7

      *           *           *
      *           *           *
    * * * * *     *           *
    * * * * *     *           *
    * * * * *     *           *
    * * * * *     *           *
    * * * * *     *           *
    * * * * *     *           *
    * * * * *     *           *

Masukan input!
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)
>> _
```

Gambar 6.14.1

Saat diminta input untuk memindahkan disk, jika input yang diberikan tidak sesuai, input tersebut akan ditolak

```
Masukan input!  
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)  
>> 13  
Input tidak valid!  
  
Masukan input!  
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)  
>> 1 3 4  
Input tidak valid!  
  
Masukan input!  
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)  
>> 1, 3  
Input tidak valid!
```

Gambar 6.14.2

Jika input valid (dari segi format input), program akan melakukan perintah sesuai input. Contoh pada Gambar 6.14.3, jika inputnya adalah “1 3” program akan memindahkan disk teratas pada tower 1 ke tower 3, dan banyaknya moves bertambah.

```
Masukan input!
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)
>> 1 3

Moves: 1
Minimum moves to solve: 7

      *           *
      *           *
      *           *
      *           *
*****          *
*****          *
*****          *
*****          *
*****          *
```

Gambar 6.14.3

Jika input tidak valid (dari segi aturan bermain), program akan mengeluarkan pesan eror dan meminta input kembali.

```
Masukan input!
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)
>> 1 3
Gagal memindahkan! Ukuran disk harus lebih kecil daripada disk di bawahnya.

Moves: 1
Minimum moves to solve: 7

      *           *           *
      *           *           *
      *           *           *
      *           *           *
*****          *           *
*****          *           *
*****          *           *
*****          *           *
*****          *           *
*****          *           *
*****          *           *
*****          *           *

Masukan input!
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)
>>
```

Gambar 6.14.4

Contoh pada Gambar 6.14.5 adalah testing jika memindahkan disk dari 1 ke 2. Hasilnya sesuai harapan karena seharusnya disk pada tower 1 akan berpindah ke tower 2.

```
Masukan input!
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)
>> 1 2

Moves: 2
Minimum moves to solve: 7

      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
*****          *****          *****
*****          *****          *****
```

Gambar 6.14.5

Kemudian, akan dilakukan testing akhir permainan. Permainan seharusnya berakhir ketika semua disk dipindahkan ke tower 3 dengan aturan yang berlaku. Gambar 6.14.6 merupakan contoh akhir dari permainan. Saat permainan berakhir program akan menampilkan skor sesuai dengan jumlah moves yang dilakukan. *Full score* jika jumlah moves sama dengan jumlah minimal *moves*.

```
Moves: 6
Minimum moves to solve: 7

      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
*****          *****          *****
*****          *****          *****

Masukan input!
Contoh input: 1 3 (Memindahkan disk teratas tower 1 ke tower 3)
>> 1 3

Moves: 7
Minimum moves to solve: 7

      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
      *           *           *
*****          *****          *****
*****          *****          *****

Berhasil! Kamu menyelesaikan dalam 7 langkah.
Skor: 100
```

Gambar 6.14.6

6.15 Tictactoe

Pada data test ini akan dilakukan pengetesan pada seluruh fitur permainan Tictactoe. Saat permainan dimulai, dicetak sebuah matriks kosong yang merupakan tempat bermain tictactoe beserta keterangan bahwa pemain 1 dinyatakan dengan “X” dan pemain 2 dengan “O”.

Pemain 1 akan dipersilahkan untuk memilih terlebih dahulu, dimana pada kasus ini pemain 1 memilih nomor 1 sehingga akan muncul “X” pada petak nomor 1.

```
Tic Tac Toe
Pemain 1 (X) - Pemain 2 (O)
1 | 2 | 3
- - -
4 | 5 | 6
- - -
7 | 8 | 9

Player 1, silakan memilih angka: 1

Tic Tac Toe
Pemain 1 (X) - Pemain 2 (O)
X | 2 | 3
- - -
4 | 5 | 6
- - -
7 | 8 | 9

Player 2, silakan memilih angka:
```

Gambar 6.15.1

Dilanjutkan dengan pemain 2 yang menginput 4 sehingga muncul “O” pada petak nomor 4.

```
Tic Tac Toe
Pemain 1 (X) - Pemain 2 (O)
X | 2 | 3
- - -
O | 5 | 6
- - -
7 | 8 | 9

Player 2, silakan memilih angka: 4
```

Gambar 6.15.2

Pada kasus ini, pemain 1 memilih kembali petak nomor 1 yang sebenarnya sudah dipilih olehnya pada kesempatan sebelumnya. Dengan begitu, masukan tidak valid dan diminta masukan kembali untuk pemain 1.

```
Tic Tac Toe
Pemain 1 (X) - Pemain 2 (O)

X | 2 | 3
---|---|---
0 | 5 | 6
---|---|---
7 | 8 | 9

Player 1, silakan memilih angka: 1
Input tidak valid
Player 1, silakan memilih ulang angka:
```

Gambar 6.15.3

Setelah pemain 1 memilih nomor 2, pada kasus ini pemain 2 memilih nomor 2 kembali dimana petak tersebut sudah dipilih oleh pemain 1. Dengan begitu, masukan tidak valid dan akan diminta masukan kembali untuk pemain nomor 2.

```
Tic Tac Toe
Pemain 1 (X) - Pemain 2 (O)

X | X | 3
---|---|---
0 | 5 | 6
---|---|---
7 | 8 | 9

Player 2, silakan memilih angka: 2
Input tidak valid
Player 2, silakan memilih ulang angka:
```

Gambar 6.15.4

Setelah input tidak valid, pemain nomor 2 memilih untuk petak nomor 5, maka petak nomor 5 akan berisi “X”

```

Player 2, silakan memilih angka: 2
Input tidak valid
Player 2, silakan memilih ulang angka: 5

Tic Tac Toe
Pemain 1 (X) - Pemain 2 (O)

  X |  X |  3
  ---|---|---
  0 |  0 |  6
  ---|---|---
  7 |  8 |  9

```

Gambar 6.15.5

Setelah pemain 2, maka akan diberi kesempatan untuk pemain 1 memilih kembali, dimana pada kesempatan kali ini pemain 1 memilih nomor 3. Seharusnya, karena petak 1, 2, dan 3 berisi X program akan berakhir dan permainan dimenangkan oleh pemain 1.

<pre> Tic Tac Toe Pemain 1 (X) - Pemain 2 (O) X X 3 --- --- --- 0 0 6 --- --- --- 7 8 9 Player 1, silakan memilih angka: 3 </pre>	<pre> Tic Tac Toe Pemain 1 (X) - Pemain 2 (O) X X X --- --- --- 0 0 6 --- --- --- 7 8 9 ==> Player 1 menang </pre>
---	---

Gambar 6.15.6 dan Gambar 6.15.7

6.16 RNG

RNG adalah permainan menebak angka random 0-100. Saat program dimulai, program akan menampilkan kalimat “RNG Telah dimulai. Uji keberuntungan Anda dengan menebak X” dan pemain akan diminta memasukkan angka tebakan. Jika angka tebakan terlalu besar maka program akan menampilkan pesan “Lebih kecil” dan jika angka tebakan terlalu kecil maka program akan menampilkan pesan “Lebih besar”. Program akan terus berjalan sampai pemain menebak dengan benar. jika pemain menebak dengan benar maka program akan menampilkan pesan “Ya, X adalah (angka tebakan pemain).”.

```

=====
RNG Telah dimulai. Uji keberuntungan Anda dengan menebak X.
Tebakan: 1
Lebih besar
Tebakan: 100
Lebih kecil
Tebakan: 50
Lebih besar
Tebakan: 75
Lebih kecil
Tebakan: 60

Ya, X adalah 60.

```

Gambar 6.16.1

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	START	untuk menguji apakah command START berfungsi dengan baik	menjalankan program, memasukkan perintah START	Data test 6.1	Mulai menjalankan program utama	Sesuai yang diharapkan
2	LOAD	untuk menguji apakah fungsi LOAD berfungsi dengan baik	menjalankan program, memasukkan perintah LOAD test.txt	Data test 6.11	Berhasil mengambil data dari savefile.txt	Sesuai yang diharapkan
3	SAVE	untuk menguji apakah command SAVE berfungsi dengan baik	menjalankan program, memasukkan perintah START, menyimpan state game pemain dengan command SAVE test.txt	Data test 6.9	Menyimpan LIST GAME terakhir yang ke file test.txt	Sesuai yang diharapkan
4	CREATE GAME	untuk menguji apakah command CREATE GAME berfungsi dengan baik	menjalankan program, memasukkan perintah START, membuat game baru dengan perintah	Data test 6.3	Membuat game baru pada LIST GAME dengan nama "Bomb"	Sesuai yang diharapkan

			CREATE GAME, menamakan game dengan nama Bomb			
5	LIST GAME	untuk menguji apakah command LIST GAME berfungsi dengan baik	menjalankan program, memasukkan perintah START, menampilkan daftar game yang tersedia dengan perintah LIST GAME	Data test 6.2	Menampilkan LIST GAME yang sedang tersimpan pada arrGame	Sesuai yang diharapkan
6	DELETE GAME	untuk menguji apakah command DELETE GAME berfungsi dengan baik	menjalankan program, memasukkan perintah START, menghapus game dengan perintah DELETE GAME, memilih game no 6 untuk dihapus	Data test 6.7	Menghapus game sesuai dengan nomor input, kecuali nomor input merupakan game default	Sesuai yang diharapkan
7	QUEUE GAME	untuk menguji apakah command QUEUE GAME berfungsi dengan baik	menjalankan program, memasukkan perintah START, memilih dan memasukkan game ke dalam antrian dengan perintah QUEUE GAME 1	Data test 6.4	Menampilkan antrian game saat ini dan memasukan game ke list antrian game yang akan di play	Sesuai yang diharapkan
8	PLAY GAME	untuk menguji apakah command PLAY GAME berfungsi dengan baik	menjalankan program, memasukkan perintah START, mulai memainkan game sesuai urutan antrian game dengan perintah PLAY GAME	Data test 6.5	Memainkan game sesuai dengan urutan QUEUE GAME	Sesuai yang diharapkan
9	SKIP GAME	untuk menguji apakah command SKIP GAME berfungsi dengan baik	menjalankan program, memasukkan perintah START, memasukkan perintah SKIP GAME 1 untuk melewati game	Data test 6.6	Melewati antrian game sesuai input, lalu memainkan game tersebut	Sesuai yang diharapkan
10	QUIT	untuk menguji apakah command QUIT	menjalankan program, memasukkan perintah START,	Data test 6.10	Keluar dari program BNMO	Sesuai yang diharapkan

		berfungsi dengan baik	memasukkan perintah QUIT			
11	HELP	untuk menguji apakah command HELP berfungsi dengan baik	menjalankan program, memasukkan perintah START, menampilkan kegunaan dari tiap command dengan perintah HELP	Data test 6.8	Menampilkan masing-masing kegunaan dari fungsi	Sesuai yang diharapkan
12	COMMAND LAIN	untuk menguji apakah command yang tidak terdefinisi tidak bekerja	menjalankan program, memasukkan perintah START, pengguna memasukan command yang salah "MAIN GAME"	Data test 6.12	Menampilkan keterangan bahwa <i>command</i> tidak dikenali dan pengguna diminta untuk memasukan <i>command</i> yang valid	Sesuai yang diharapkan
			menjalankan program, memasukkan perintah LOAD coba.txt	Data test 6.11	Menampilkan keterangan bahwa file tidak dapat dibaca karena tidak ditemukan file tersebut pada folder "data". Program akan meminta masukan kembali.	Sesuai yang diharapkan
13.	DINER DASH	untuk memainkan game dinner dash	menjalankan program dengan perintah START memasukkan game diner dash dengan perintah QUEUE GAME memainkan game dengan perintah PLAY GAME memainkan game dengan menggunakan perintah COOK dengan satu parameter berupa ID makanan untuk memasak	Data test 6.13	pemain kalah dan hanya berhasil memasak dan mengantarkan 1 makanan ke pelanggan	Sesuai yang diharapkan

			atau perintah SERVE dengan satu parameter berupa ID makanan untuk menyajikan makanan atau SKIP untuk melewati satu putaran tanpa melakukan apa-apa			
14.	Tower of Hanoi	untuk menguji apakah permainan Tower of Hanoi berjalan dengan baik	menjalankan program, memasukkan perintah START, memasukkan permainan Tower of Hanoi ke dalam antrian dengan perintah QUEUE GAME, memulai permainan dengan perintah PLAY GAME, memainkan Tower of Hanoi dengan memasukkan perintah berupa 2 digit angka 1-3 yang berbeda untuk memindahkan disk	Data test 6.14	Disk berhasil di urutkan berdasarkan paling besar dari bawah	sesuai dengan yang diharapkan
15.	Tictactoe	untuk menguji apakah permainan tictactoe berjalan dengan baik	menjalankan program, memasukkan perintah START, memasukkan game Tictactoe ke dalam antrian dengan perintah QUEUE GAME, memulai permainan dengan perintah PLAYGAME, memainkan tictactoe dengan memasukkan perintah berupa angka 1-9	Data test 6.15	pemain 1 menang dan permainan tetap berjalan meskipun perintah yang dimasukkan tidak valid	Sesuai yang diharapkan
16.	RNG	untuk menguji apakah permainan RNG berjalan dengan baik	menjalankan program, memasukkan perintah START, memasukkan game RNG ke dalam	Data test 6.16	menampilkan pesan “Lebih besar” ketika pemain menginput tebakan yang	sesuai yang diharapkan

			antrian dengan perintah QUEUE GAME, memulai permainan dengan perintah PLAY GAME, memainkan RNG dengan memasukkan perintah berupa angka 0-100		terlalu kecil, menampilkan pesan “Lebih kecil” ketika pemain memasukkan angka yang terlalu besar dan menampilkan pesan “Ya, X adalah (angka tebakan pemain).” ketika pemain berhasil menebak	
--	--	--	--	--	--	--

8 Pembagian Kerja dalam Kelompok

NIM	Nama	Pembagian Kerja
18221162	Ceavin Rufus De Prayer Purba	<ul style="list-style-type: none"> - Membuat implementasi permainan Diner Dash, Tower of Hanoi, serta optimalisasi permainan Tictactoe menjadi menggunakan ADT matriks. - Membuat ADT serta driver untuk keperluan implementasi yang telah disebutkan di atas (ADT stack, mesin kata, mesin karakter, matrix, order, masakan, dan array) - Debugging dan fixing seluruh bagian program untuk mencapai kebutuhan sesuai spek - Menambahkan/merevisi laporan di bagian algoritma unik, data test, penjelasan ADT, dan ringkasan - Mengerjakan data test bagian Tower of Hanoi
18221118	Ardhan Nur Urfan	<ul style="list-style-type: none"> - Membuat dan menggabungkan procedure ke dalam main program (masing-masing console)

		<ul style="list-style-type: none"> - Membuat prosedur Load Game, Create Game, List Game, Delete Game, Play Game - Memodifikasi ADT mesin karakter, ADT mesin kata, dan ADT arraydin - Membuat beberapa fungsi dan prosedur yang mendukung implementasi
18221046	Vincent Winarta	<ul style="list-style-type: none"> - Membuat function SKIP GAME - Membuat function QUEUE GAME - Melakukan data testing untuk main function pada laporan - Mengerjakan data script pada main function
18219048	Dwiky Hared Darmawan	<ul style="list-style-type: none"> - Membuat ringkasan - Mengedit laporan bagian program utama - Melakukan Data testing pada program RNG - Mengerjakan Data script pada program RNG - Mengedit test script
18221084	Rei Arriel Clyfton	<ul style="list-style-type: none"> - Membuat implementasi rng, dengan time.h dan tanpa time.h - Membuat implementasi bonus game (Tictactoe) - Membuat ADT array untuk Tictactoe (sebelum dioptimalisasi) - Formatting laporan
18221128	Fadhlán Nazhif Azizy	<ul style="list-style-type: none"> - Membuat procedure SAVE - Melakukan data testing pada game diner dash - Mengisi hasil data testing pada bagian data script diner dash - Mencatat hasil asistensi

9 Lampiran

9.1 Deskripsi Tugas Besar 2

BNMO (dibaca: Binomo) adalah sebuah robot video game console yang dimiliki oleh Indra dan Doni. Dua bulan yang lalu, ia mengalami kerusakan dan telah berhasil diperbaiki. Sayangnya, setelah diperbaiki ia justru mendapatkan lebih banyak bug dalam sistemnya. Oleh karena itu, Indra dan Doni mencari programmer lain yang lebih handal untuk ulang memprogram robot video game console kesayangannya.

1. About the System

BNMO merupakan suatu robot game console yang dapat menjalankan permainan. BNMO memiliki beberapa fitur utama, yaitu:

- Memainkan game
- Menambahkan game
- Menghapus game
- Mengurutkan game yang akan dimainkan

2. Main Menu

Ketika program pertama kali dijalankan, BNMO akan memperlihatkan main menu yang berisi welcome page dan beberapa menu pilihan yaitu START dan LOAD. Setelah itu, main menu akan menerima input commands yang akan dijelaskan pada bagian berikutnya.

3. Command

Pada setiap giliran, pemain dapat memasukkan command-command berikut:

a. START

START merupakan salah satu command yang dimasukkan pertama kali oleh pemain ke BNMO. Setelah menekan Enter, dibaca file konfigurasi default yang berisi list game yang dapat dimainkan.

b. LOAD <filename>

LOAD merupakan salah satu command yang dimasukkan pertama kali oleh pemain ke BNMO. Memiliki satu argumen yaitu filename yang merepresentasikan suatu save file yang ingin dibuka. Setelah menekan Enter, akan dibaca save file <filename> yang berisi list game yang dapat dimainkan, histori dan scoreboard game, lebih detailnya bisa dilihat pada Konfigurasi Sistem.

c. SAVE <filename>

SAVE merupakan command yang digunakan untuk menyimpan state game pemain saat ini ke dalam suatu file. Command SAVE memiliki satu argumen yang merepresentasikan nama file yang akan disimpan pada disk.

d. CREATEGAME

CREATEGAME merupakan command yang digunakan untuk menambahkan game baru pada daftar game. Spesifikasi game yang dibuat dapat dilihat pada section Spesifikasi Game

e. LISTGAME

LISTGAME merupakan command yang digunakan untuk menampilkan daftar game yang disediakan oleh sistem.

f. DELETGAME

DELETGAME merupakan command yang digunakan untuk menghapus sebuah game dari daftar game. Adapun aturan penghapusan game adalah:

Game yang dapat dihapus hanya game yang dibuat secara custom oleh pengguna.

5 game pertama pada file konfigurasi tidak dapat dihapus.

Game yang saat itu terdapat di dalam queue game tidak dapat dihapus.

g. QUEUEGAME

QUEUEGAME merupakan command yang digunakan untuk mendaftarkan permainan kedalam list. List dalam queue akan hilang ketika pemain menjalankan command QUIT.

h. PLAYGAME

PLAY GAME merupakan command yang digunakan untuk memainkan sebuah permainan. Game yang dimainkan adalah game dengan urutan pertama di antrian game. Ketika salah satu permainan dimulai, sistem akan menjalankan game sesuai pada section Spesifikasi Game. Permainan selain yang dispesifikasikan pada Spesifikasi Game akan menampilkan pesan bahwa game tidak dapat dimainkan.

i. SKIPGAME <n>

SKIPGAME merupakan command yang digunakan untuk melewati permainan sebanyak n.

j. QUIT

Keluar dari program.

k. HELP

Bantuan command-command yang disebutkan di atas. Tampilan dan kata-kata dibebaskan.

l. COMMAND LAIN

Command-command lain selain yang disebutkan diatas tidak valid. Keluar dari program.

9.2 Notulen Rapat

Rapat 1	Catatan Rapat: Pembagian tugas: <ol style="list-style-type: none"> 1. Ceavin: Diner Dash 2. Rei: RNG & Game bonus 3. Ardhan, Alan, Vincent: Command
Tanggal : 29 Oktober 2022	
Tempat : LINE	

STEI- ITB	IF2111 TBI 13	Halaman 34 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

Kehadiran Anggota Kelompok: 18221162 Ceavin Rufus De Prayer Purba 18221084 Rei Arriel Clyfton 18221046 Vincent Winarta 18219048 Dwiky Hared Darmawan 18221118 Ardhan Nur Urfan 18221128 Fadhlan Nazhif Azizy	4. Dwiky: Laporan Pembagian nanti fleksibel tergantung ke depannya gimana. Trus kalau mau ada yang bikin game bonus selain Rei nanti boleh buat aja
---	--

9.3 Log Activity Anggota Kelompok

Keterangan lebih lanjut mengenai log activity saat membuat program dapat dilihat di [.Commit History](#)

Log Act	Anggota kelompok yang terlibat	Tanggal
Pembagian tugas	18219048 Dwiky Hared Darmawan 18221162 Ceavin Rufus De Prayer Purba 18221046 Vincent Winarta 18221084 Rei Arriel Clyfton 18221118 Ardhan Nur Urfan 18221128 Fadhlan Nazhif Azizy	29 Oktober 2022
Membuat program Diner Dash	18221162 Ceavin Rufus De Prayer Purba	30 Oktober - 1 November 2022
Membuat program RNG	18221084 Rei Arriel Clyfton	30 Oktober 2022
Mulai mengerjakan prosedur-prosedur pada console.c (perintah di main program)	18221118 Ardhan Nur Urfan 18221128 Fadhlan Nazhif Azizy 18221046 Vincent Winarta	1 November 2022
Membuat program TicTacToe	18221084 Rei Arriel Clyfton	3 - 4 November 2022
Mulai pembuatan laporan	18219048 Dwiky Hared Darmawan 18221046 Vincent Winarta	7 November 2022
Membuat program Tower of Hanoi	18221162 Ceavin Rufus De Prayer Purba	9 November 2022
Debugging & fixing	18221162 Ceavin Rufus De	9 - 11 November 2022

	Prayer Purba 18221046 Vincent Winarta 18221118 Ardhan Nur Urfan 18221128 Fadhlan Nazhif Azizy	
Finalisasi laporan	18221162 Ceavin Rufus De Prayer Purba 18221046 Vincent Winarta 18219048 Dwiky Hared Darmawan	11 November 2022

9.4 MoM Asistensi Tugas Besar



MoM asistensi tugas besar terlampir pada bagian akhir laporan.

**Form Asistensi Tugas Besar
IF2110/Algoritma dan Struktur Data
Sem. 1 2022/2023**

No. Kelompok/Kelas : 13/K02
 Nama Kelompok : 13INOMCUK
 Anggota Kelompok (Nama/NIM) :
 1. Dwiky Hared Darmawan/18219048
 2. Ceavin Rufus De Prayer Purba / 18221162
 3. Vincent Winarta / 18221046
 4. Rei Arriel Clyfton / 18221084
 5. Ardhan Nur Urfan / 18221118
 6. Fadhlán Nazhif Azizy / 18221128

Asisten Pembimbing : Aditya Bimawan /13519064

Asistensi I

Tanggal : 4 November 2022	Catatan Asistensi: So far sudah bagus karena harusnya tinggal gabungin function-function dengan gamenya. kendala jalanin game pake ./main gak bsa jadi harus execute program. Buat nge run gak jadi masalah asalkan cara nge run nya ditulis. asisten ngeceknya ikutin yang ditulis itu. Disarankan menambahkan try except/catch. Tidak perlu menambah library untuk kepentingan tampilan, mengantisipasi asisten yang melakukan asistensi dan menilai itu beda. Error kalau save / load harus dihandle untuk ngasih tau ke user kalau command-nya salah.
Tempat : Zoom Meeting	
Kehadiran Anggota Kelompok: 1 18221046 Vincent Winarta  2 18221162 Ceavin Rufus De Prayer Purba  3	

18221084
Rei Arriel Clyfton



4
18219048
Dwiky Hared Darmawan



5
18221118
Ardhan Nur Urfan







6
18221128
Fadhlan Nazhif Azizy





Tanda Tangan Asisten:



Asistensi II

Tanggal : 9 November 2022	Catatan Asistensi: RNG tidak harus memakai library time.h Bagian output yang diharapkan itu disesuaikan dengan keinginan / harapan kita. Kalau ada fitur yang belum selesai ditampilkan disini harapan outputnya. Harus ada driver untuk sebagai test case buat semua adt. ADT order coba dijelasin aja kenapa dimodifikasi seperti itu Game bonus langsung taro di file .txt aja, gaperlu pake create game biar ada di list Data test diisi dengan skenario pengetesan yang dibikin sendiri. Pada bagian data testing, masukan input-input testing ke dalam penjelasan Pada bagian test script, hasil yang keluar bisa jadi sesuai dengan hasil yang diharapkan jika semua function dapat bekerja dengan baik Program utama diceritain main programnya itu gimana jalannya Penjelasan ADT gaperlu ditulis fungsi fungsi apa aja yang ada di dalemnya
Tempat : Zoom Meeting	
Kehadiran Anggota Kelompok:	
1 18221046 Vincent Winarta  2 18221162 Ceavin Rufus De Prayer Purba  3 18221084 Rei Arriel Clyfton  4 18219048 Dwiky Hared Darmawan 	

<p>5 18221118 Ardhan Nur Urfan</p>  <p>6 18221128 Fadhlan Nazhif Azizy</p> 	
	<p>Tanda Tangan Asisten:</p> 