CHAPTER V
IMPLEMENTATION AND TESTING

5.1 Implementation

Execute the c file requires gcc compiler, first make sure gcc is already installed on linux terminal. Then compile the project file with command :

gcc -o uno uno.c

Then running the output compiler with :

./uno

Option menu will immediately appeared, then choose 1 to fill the username. Afterwards choose 2 to shuffle the deck, then 3 to deals the cards to players and later choose play to start the game. Below are explanation of important step :

5.1.1 Read the file

```
FILE *fp;

char* slice;
char fetch[15];

if ((fp=fopen("kartu.txt","r"))==NULL){
    printf("Kesalahan! File tidak ditemukan!!\n");
    exit(1);
}

ptr = (node *)malloc(sizeof(node));

while(fscanf(fp,"%[^\n]",fetch)!=EOF){
    fscanf(fp,"%*c");
    head = (node *)malloc(sizeof(node));
    //get angka
    slice = strtok(fetch," ");
    strcpy(head->angka, slice);

    //get warna
    slice = strtok(NULL," ");
    strcpy(head->warna, slice);

    head->next=ptr;
    ptr=head;
}
```

Figure 5.1 Read File

Uno card has 2 main information, which are number / symbols and color. Both information are read and record separately, strtok used to split the information.

5.1.2 Shuffle Deck

Shuffling is done by two step, first is shuffle function. This function will be randoming number on 2 variables (for the next will be call variable a and b) which is from 0 until 107, then the numbers will be send to swap function. This will be looping for the length of the deck which is 108 times.

```
node* shuffle(node* head){

    deck = (node *)malloc(sizeof(node));

    int i = 0;
    int randomNumb = 0;
    int randomNumb2 = 0;
    deck=head;

    for(i=0;i<length(head);i++){
        //randoming 2 different number
        randomNumb = 1 + (rand()%length(head));
        randomNumb2 = 1 + (rand()%length(head));
        deck= swap(deck,randomNumb,randomNumb2);
    }

    //delete last next null
    deck=delete(deck);
}
```

Figure 5.2 Shuffle Function

In swap function, these random numbers on variable a or b will be used to call node according the number. Get index function is used to help get the right node. Example : number 15 will call node on linked list sequence to 15.

```
node* getIndex(int index,node *theList){
int size = 0;

    //while list's content is not null, looping to next list
    while(theList!=NULL){
    if(size == index) return theList;

    size = size + 1;
    theList = theList->next;
    }

}
```

Figure 5.3 Get Index Function

Swap function will be running such as :

- Variable a will contain a sequence nodes according to the results of random number b, vice versa
- Swap function has counter variable, this variable used temporary as pointer to sequences node.
- If counter and variable a nor b doesn't match then insert node normally to the linked list then return the value to the shuffle function
- When counter found match with one of these variables, then check the variable. If counter match with variable a then the node will contain value node from variable b, vice versa. Then insert the node normally to the linked list then return the value to the shuffle function.

```
node* swap(node* theList,int randomNumb,int randomNumb2){

node* fetchNode=getIndex(randomNumb,theList);
node* fetchNode2=getIndex(randomNumb2,theList);
node *result, *ptr;

    int size = 0;
    int maxsize = length(theList);

    ptr = (node *)malloc(sizeof(node));
    if(randomNumb != randomNumb2){
        for(theList; theList!=NULL; theList=theList->next){

            if(size<108){
            result = (node *)malloc(sizeof(node));

                if(size == randomNumb){
                strcpy(result->warna,fetchNode2->warna);
                strcpy(result->angka,fetchNode2->angka);
                }
                else if(size == randomNumb2){
                strcpy(result->warna,fetchNode->warna);
                strcpy(result->angka,fetchNode->angka);
                }
                else{
                strcpy(result->warna,theList->warna);
                strcpy(result->angka,theList->angka);
                }

            result->next=ptr;
            ptr=result;
```

Figure 5.4 Swap Function

5.1.3 Deal Card

At beginning of the game every player must have 7 cards. Set 5 separately new linked list for 4 players then insert that 4 linked list with 7 node. The last linked list will be insert with the rest of the node and this linked list will be deck.

Plot of deal card

- Set 5 new linked list, for 4 player use struct from player and deck use node
- Set the counter for count the card, every multiple 7 then insert 7 node to player struct and this will be done until the fourth player.
- If all player already have cards the the rest of the card will be insert to the deck linked list

5.1.4 Uno Game

The uno game use the principle of the greedy algorithm. It will be throw the card from player hand which has highest rank first, if all card has same rank then the throw card will be thrown in accordance with normal sequences.

Plot of Game

Below are sequences in the play function :

count card deck / players : integer{check the status of the game}

check first card : string{check for activate flag / normal}

check reverse : boolean { check the rotation of game}

kirimbuang : string {check the last discard card}

find : integer, string, boolean {call the algorithm}

show (tempbuang) : string {display the last discard card}

uno : string {check if the player card only one}

Game

Below are the pseudo code for the game :

```
while(boolean flag) -> check the flag
    {
    if (bool draw 4) ->active
        then goto draw 4
    else if (bool draw 2)->active
        then goto draw 2
    else if (bool skip)->active
        then goto skip
    else if (normal) ->active
        then goto normal
    }

    skip :
    read the array
    {
            do the calculation
            set the flag
    }

    draw 4:
    {
    if(has card)
        do throw card and choose color
        set the flag
    else
        draw cards from deck
        set the flag
    }


    draw 2 :
    {
    if(has card)
        do throw card
        set the flag
    else
        draw cards from deck
        set the flag
    }

    normal :
    {
    if(has card)
        if(Draw 4)
        {
            do throw card and choose color
            set the flag
        }
        else if(Wild)
        {
            do throw card and choose color
            set the flag
        }
        else
        do throw card
        set the flag
    else
        draw cards from deck
        set the flag
    }
```

Figures 5.5 Implementation of Uno

Element on uno game :

- Set of candidate: players cards.
- Solution candidate: check the flag
- Selection function: choose the decision
- Feasibilty function: checker number or symbol and color from active card
- Objective function: card to be played are high priority

5.2 Testing

To know the program work properly, simulation of the game is necessary. Below are some result from simulation :

```
root@alvin-K42Jc:/home/alvin/Desktop/uno# gcc -o uno uno.c
root@alvin-K42Jc:/home/alvin/Desktop/uno# ./uno
--------------------UNO-----------------
1.      Player Name
2.      Shuffle
3.      Setup Player
4.      Play
5.      Quit

Make your selection:
```

Figure 5.6 Menu Interface

Then result of the simulation of uno, game is start with open card / dicard card from the deck. Below are the result of the simulation. The open card is reverse mean the game will play anticlockwise, that means player 3 will be start first then following by player 2 and so on until found reverse card again to play game normally.

Computer cards aren't displayed, it will be display only the number of cards and information what computer throw.

```
kartu bukaan awal :
«» merah
bukaan skrg: «» merah

giliran pemain ke 3
jumlah kartu pemain ke 3 adalah 6
warna sama


«» merah
1 merah

bukaan skrg: 1 merah

giliran pemain ke 2
jumlah kartu pemain ke 2 adalah 6
warna sama


«» merah
1 merah
0 merah

bukaan skrg: 0 merah
```

Figure 5.7 Reverse Card on Computer Turn

User interface on user has index, the index will be use as the input to throw the card. Hint is provided to assist user, there are some information about the card.

```
bukaan skrg: 9 merah

giliran alvin
======kartu alvin========
0   5 kuning
1   +2 kuning
2   7 biru
3   5 biru
4   8 kuning
5   D+4 hitam
6   8 kuning

Hint tekan 98
Masukan nilai index untuk normal : 98
dapat mengeluarkan draw 4

Hint tekan 98
Masukan nilai index untuk normal :
```

Figure 5.8 Normal User Turn

When draw 2 card on the open card and the effect are active, it will be counting the previously draw 2 card which is active.

```
bukaan skrg: +2 hijau

giliran pemain ke 2
jumlah kartu pemain ke 2 adalah 6
mengeluarkan draw 2


5 hijau
8 hijau
+2 hijau
+2 hijau

bukaan skrg: +2 hijau

giliran pemain ke 3
jumlah kartu pemain ke 3 adalah 6
ambil minuman 2
jumlah kartu yg diambil : 4
```

Figure 5.9 Draw 2 Card Effect

Wild has special effect, it can be change the color. It can be throw anytime or if player don't have same color like the open card. Then, draw 4 card it also has effect like draw 2 card. The different are sum of the card if the effect is active which is 4 cards also it can choose the color like wild card

```
giliran alvin
======kartu alvin========
0   D+4 hitam
1   9 kuning
2   W hitam
3   4 kuning
4   6 kuning
5   5 kuning

Hint tekan 98
Masukan nilai index untuk normal : 98
dapat mengeluarkan draw 4
dapat mengeluarkan wild

Hint tekan 98
Masukan nilai index untuk normal : 2
Wild keluar
0   W kuning
1   W merah
2   W hijau
3   W biru
Masukan nilai index untuk pilih warna :
```

Figure 5.10 Wild Card Effect

Skip card has effect to skip the next player turn. If the player 1 throw skip card then player 2 will lose his turn and the game continue to player 3.

```
bukaan skrg: 0 biru

giliran pemain ke 3
jumlah kartu pemain ke 3 adalah 3
skip biasa
```

Figure 5.11 Skip Card on Computer Turn