LAPORAN TUGAS BESAR I

PROGRAM *SELF-ORDERING MACHINE*

Ditujukan untuk memenuhi tugas besar

mata kuliah Pengenalan Komputasi KU1102

oleh

19623052 – Angelina Efrina Prahastaputri

19623087 – Naomi Risaka Sitorus

19623227 – Jessica Allen

19623248 – Nathan Jovial Hartono

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

BAB I

HASIL TUGAS 1

A. Topik Persoalan

Topik tugas besar Pengenalan Komputasi yang telah kami pilih adalah sistem *self-ordering* yang biasanya terdapat pada rumah makan. Sistem *self-ordering* sendiri dapat didefinisikan sebagai *e-ordering*. Dalam hal ini, pembeli dapat memesan makanan secara elektronik melalui fasilitas teknologi yang sudah disediakan oleh restoran *(E.Abel and Obeten, 2015)*.

Sistem *self-ordering* dapat dilakukan dalam berbagai bentuk. Misalnya, melalui *website* yang dapat diakses oleh pelanggan melalui *device* masing-masing, melalui aplikasi yang telah dikembangkan oleh restoran, dan lain-lain. Namun, yang paling sering digunakan adalah sistem *self-ordering* dalam bentuk mesin fisik yang dapat diakses secara langsung oleh pelanggan di lokasi restoran. Data-data yang dimasukkan oleh pelanggan ke mesin *self-ordering* akan disimpan dan kemudian diproses lebih lanjut oleh restoran.

Fungsi dari mesin *self-ordering* sendiri tentunya adalah untuk memesan makanan secara elektronik dan mandiri. Sistem *self-ordering* ini juga memiliki banyak manfaat bagi pelanggan dan restoran. Beberapa manfaat sistem *self-ordering* bagi restoran di antaranya adalah untuk mengatasi antrian pelanggan yang panjang, menekan jumlah sumber daya manusia (mengurangi pengeluaran upah karyawan), meminimalisir terjadinya kecurangan kasir serta kesalahan penyajian pesanan, dan sebagainya. Para pelanggan juga akan merasa lebih leluasa untuk menjelajahi menu dan memesan tanpa tekanan waktu dengan adanya mesin *self-ordering* ini.

Dengan fungsi dan manfaat-manfaat tersebut, tentunya sudah banyak sekali restoran atau rumah makan yang menerapkan sistem *self-ordering*,

terutama restoran cepat saji dengan tujuan utama mempersingkat waktu pemesanan. Lokasi restoran yang kami pilih dengan sistem *self-ordering* untuk melakukan survei adalah restoran cepat saji McDonald's.

B. Dekomposisi Persoalan

Setelah melakukan survei terhadap mesin *self-ordering*, kami menyimpulkan bahwa program mesin *self-ordering* umumnya terdiri atas bagian-bagian berikut.

1. Pemesanan
    a. Pemilihan Lokasi Makanan
    b. Pemilihan Kategori Menu
    c. Pemilihan Menu yang Diinginkan
    d. Konfirmasi Pemesanan
    e. Pemilihan Tempat Pengambilan Makanan
2. Pembayaran
    a. Pemilihan Tempat Pembayaran
    b. Cetak Struk

BAB II
HASIL TUGAS 2

A. Deskripsi Simulasi

Pada tugas besar Pengenalan Komputasi ini, kelompok kami telah memilih untuk membuat program *self-ordering machine*. Setelah melakukan survei dan dekomposisi terhadap topik yang telah kami pilih, kami menyadari bahwa program *self-ordering machine* memiliki dua bagian utama, yaitu bagian pemesanan dan bagian pembayaran. Kami merasa bahwa bagian pemesanan pada program *self-ordering machine* ini paling relevan dengan topik utama yang telah kami pilih dan dengan materi Python yang sedang kami pelajari. Maka dari itu, kami hanya akan memfokuskan pada bagian pemesanan dalam penyusunan program simulasi sistem *self-ordering* ini.

Program simulasi yang kami buat akan menerima input dari *user* berupa tempat makan, kategori menu yang ingin dilihat, menu makanan yang ingin dipesan, dan metode pengambilan makanan. Input yang diterima oleh program kemudian akan disimpan dan dijalankan melalui serangkaian proses, seperti perhitungan total harga dan lainnya. Program akan terus menerima input selagi *user* masih ingin menambahkan pesanannya, tanpa menghapus input yang telah diterima sebelumnya. Setelah melalui serangkaian proses, hasil yang didapatkan dari proses tersebut akan disajikan kepada *user* dalam bentuk struk pemesanan. Pada struk pemesanan, akan tertera keterangan tempat makan, menu yang dipesan beserta jumlah dan harganya, total harga yang harus dibayar, serta metode pengambilan hasil pesanan.

Hal pertama yang akan terjadi setelah menjalankan program ini adalah *user* akan diarahkan untuk memilih lokasi makan. Pilihan yang diberikan adalah makan di tempat atau dibawa pulang. *User* akan memberikan input berupa angka dari opsi yang ingin dipilih. Setelah memilih lokasi, *user* akan diarahkan ke tampilan selanjutnya untuk memilih kategori makanan yang ingin dipesan. Kategori menu yang ada pada tampilan adalah Promosi, *Burger and Nuggets*, Ayam, *Happy Meal*, Paket Keluarga, Menu Receh, Camilan, Minuman, Pencuci Mulut, dan Nasi. Kategori menu ini kami dasarkan pada kategori menu lokasi kami melakukan survei. Sesudah memilih kategori menu yang ingin dipesan, *user* akan ditampilkan menu makanan dari kategori tersebut. Untuk menambahkan menu ke dalam pesanan, *user* harus memasukkan angka pilihan yang tertera pada menu yang diinginkan.

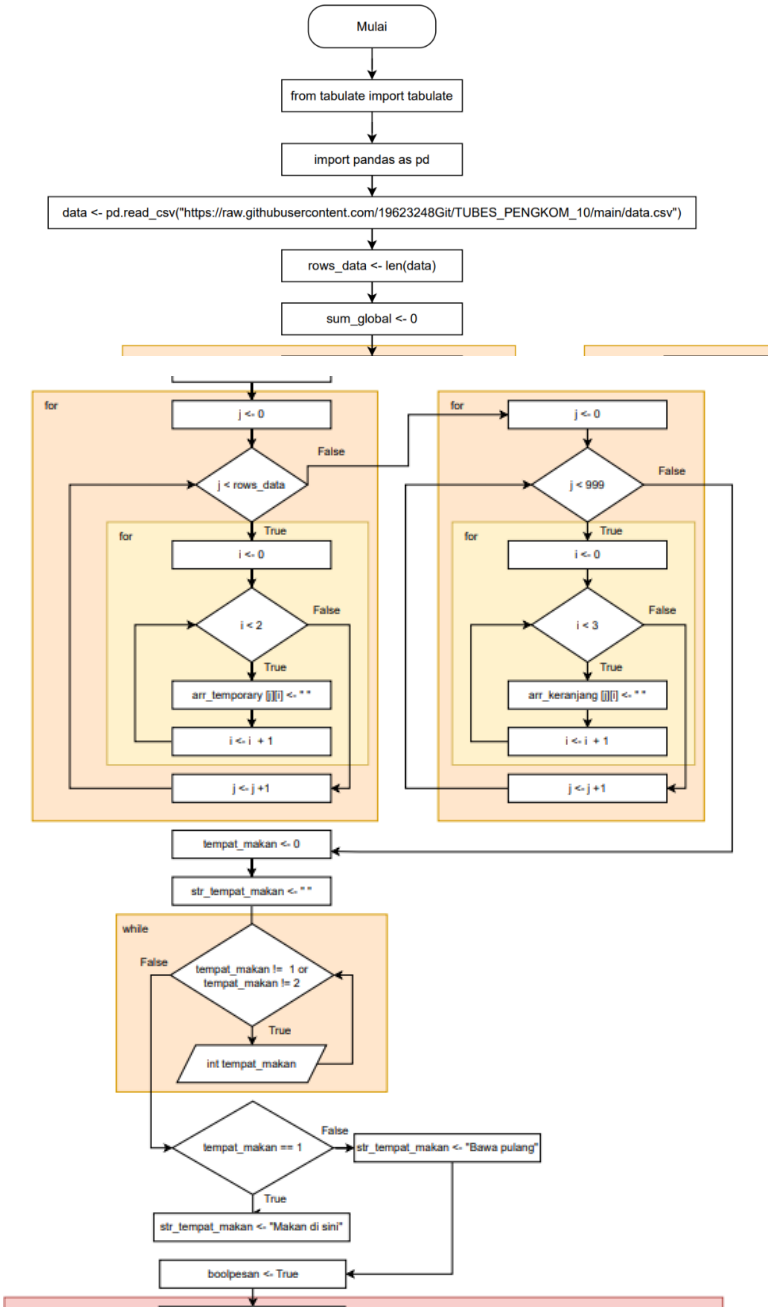
User kemudian akan diminta untuk mengkonfirmasi penambahan pesanan. Setelah itu, *user* juga akan ditanyakan apakah memiliki keinginan untuk menambah pesanannya. Jika *user* ingin menambahkan pesanannya, *user* akan diarahkan kembali ke tampilan kategori menu. Lalu, program akan berulang dari bagian tersebut. Sementara itu, program akan menyimpan pesanan-pesanan yang telah ditambahkan oleh pengguna, serta menghitung harga total dari pesanan-pesanan tersebut. Jika *user* sudah tidak ingin menambahkan pesanan, maka *user* akan langsung diarahkan untuk memilih opsi pengambilan makanan. Pilihannya adalah antara diantar ke meja atau diambil

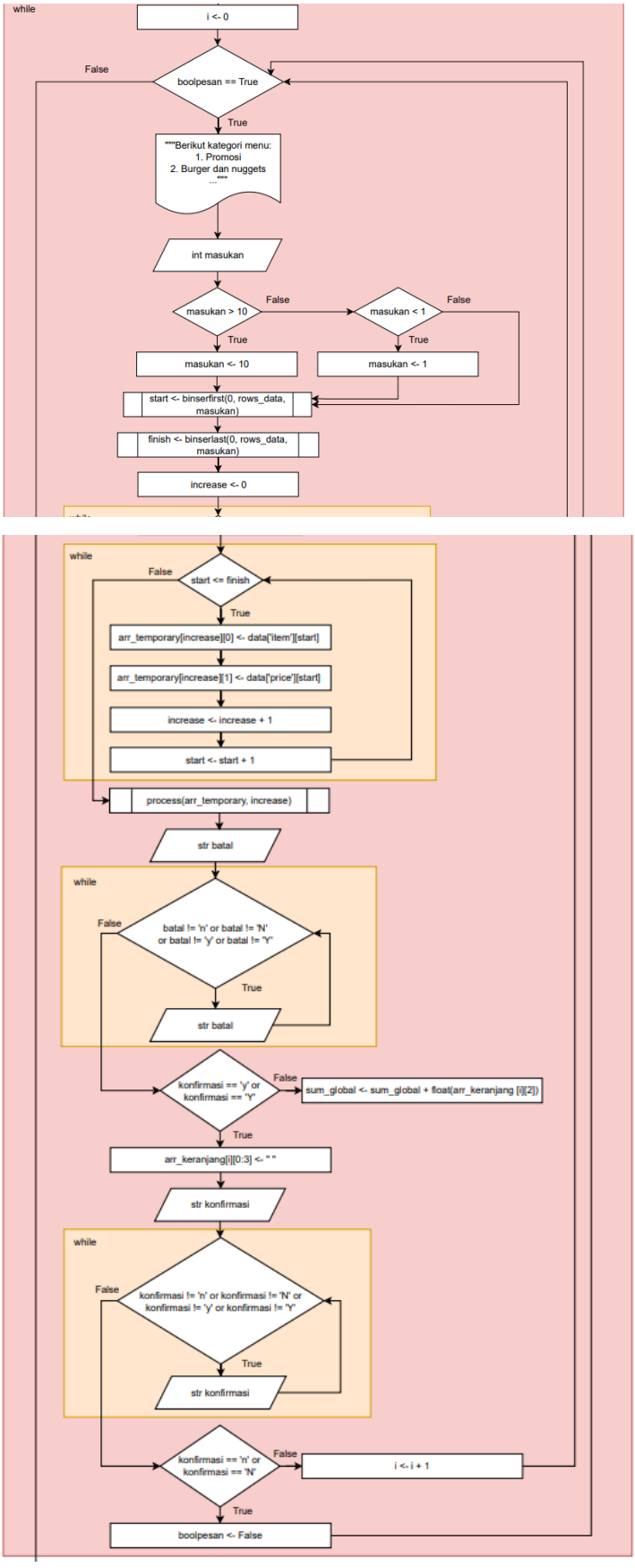
sendiri ke konter. Jika pengguna memilih untuk diantar ke meja, maka akan diminta untuk memasukkan nomor meja.

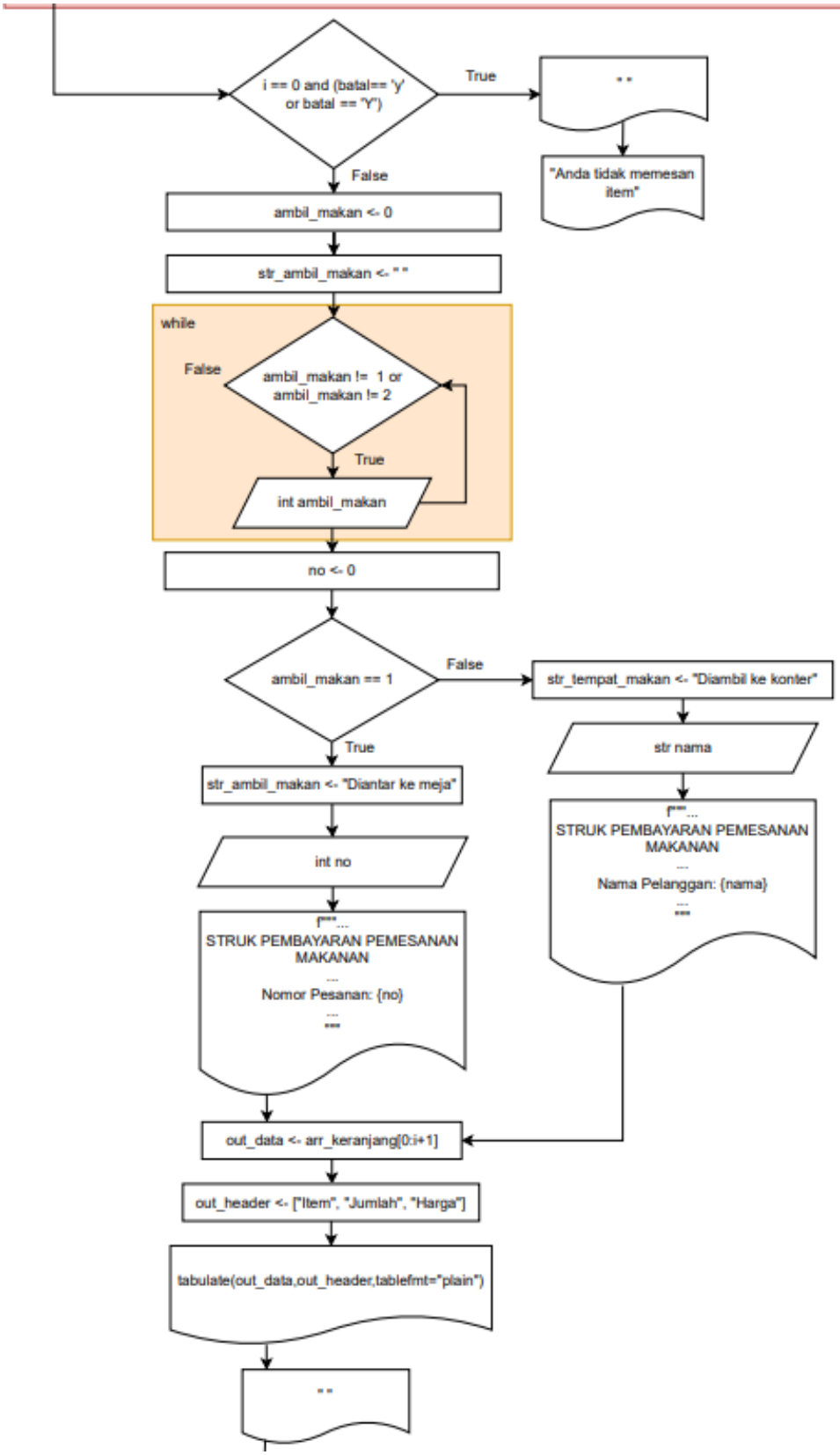
Terakhir, program akan menampilkan struk pemesanan sebagai output. Struk pemesanan akan menampilkan tempat makan, cara pengambilan makanan, daftar menu makanan yang telah dipesan beserta harganya, dan total harga pemesanan.

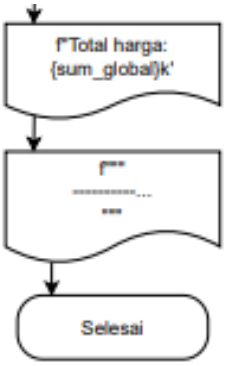
B. Flowchart

Main Flowchart:

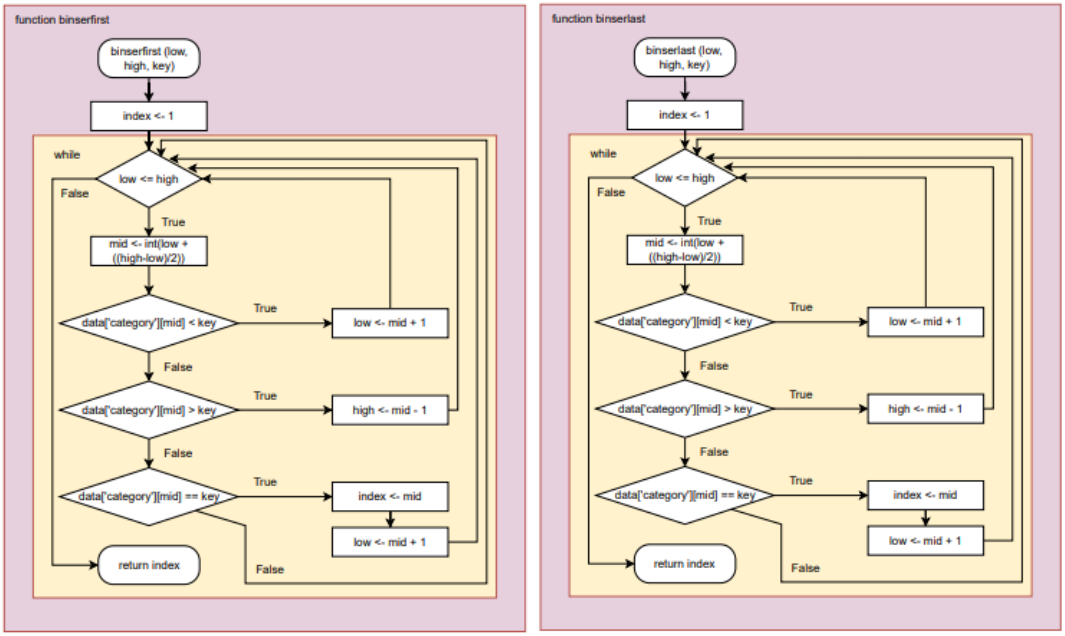


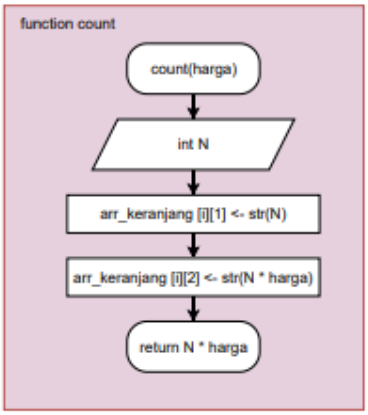

while
i <- 0
False
boolpesan == True
True
"""Berikut kategori menu:
1. Promosi
2. Burger dan nuggets
..."""
int masukan
masukan > 10
False
masukan < 1
False
True
True
masukan <- 10
masukan <- 1
start <- binserfirst(0, rows_data, masukan)
finish <- binserlast(0, rows_data, masukan)
increase <- 0
while
False
start <= finish
True
arr_temporary[increase][0] <- data['item'][start]
arr_temporary[increase][1] <- data['price'][start]
increase <- increase + 1
start <- start + 1
process(arr_temporary, increase)
str batal
while
False
batal != 'n' or batal != 'N'
or batal != 'y' or batal != 'Y'
True
str batal
konfirmasi == 'y' or
konfirmasi == 'Y'
False
sum_global <- sum_global + float(arr_keranjang [i][2])
True
arr_keranjang[i][0:3] <- " "
str konfirmasi
while
False
konfirmasi != 'n' or konfirmasi != 'N' or
konfirmasi != 'y' or konfirmasi != 'Y'
True
str konfirmasi
konfirmasi == 'n' or
konfirmasi == 'N'
False
i <- i + 1
True
boolpesan <- False

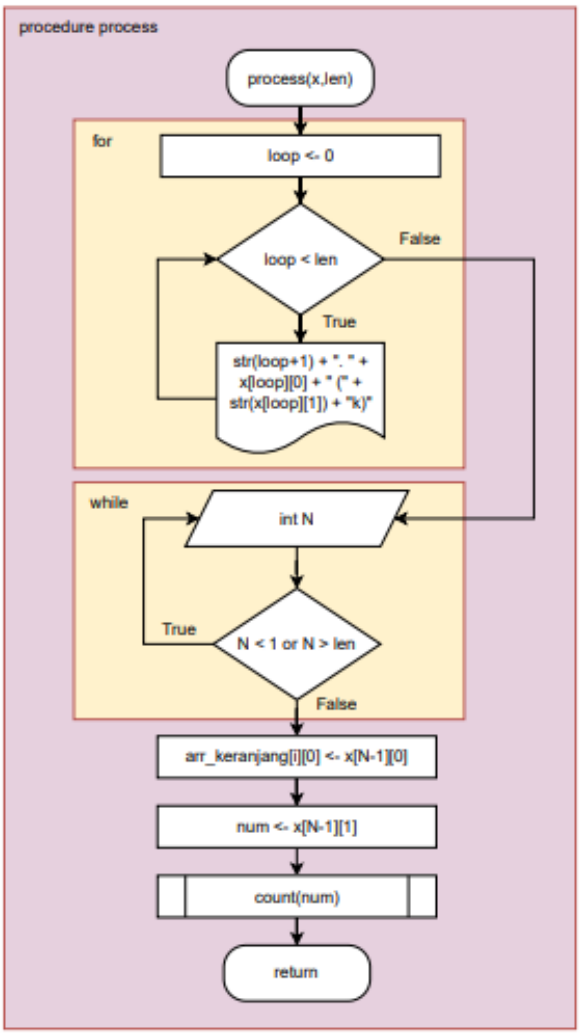

i == 0 and (batal== 'y' or batal == 'Y')
True
" "
"Anda tidak memesan item"
False
ambil_makan <- 0
str_ambil_makan <- " "
while
False
ambil_makan != 1 or ambil_makan != 2
True
int ambil_makan
no <- 0
ambil_makan == 1
False
str_tempat_makan <- "Diambil ke konter"
str nama
f"""...
STRUK PEMBAYARAN PEMESANAN MAKANAN
...
Nama Pelanggan: {nama}
...
"""
True
str_ambil_makan <- "Diantar ke meja"
int no
f"""...
STRUK PEMBAYARAN PEMESANAN MAKANAN
...
Nomor Pesanan: {no}
...
"""
out_data <- arr_keranjang[0:i+1]
out_header <- ["Item", "Jumlah", "Harga"]
tabulate(out_data,out_header,tablefmt="plain")
" "



Functions and Procedures Flowchart:






procedure process
process(x,len)
for
loop <- 0
loop < len
False
True
str(loop+1) + ". " +
x[loop][0] + " (" +
str(x[loop][1]) + "k)"
while
int N
True
N < 1 or N > len
False
arr_keranjang[i][0] <- x[N-1][0]
num <- x[N-1][1]
count(num)
return

BAB III
HASIL TUGAS 3

Dekomposisi dan *flowchart* yang telah dirancang pada tugas 2 kemudian disusun menjadi sebuah program sebagai hasil dari tugas 3. Program tersebut disusun menggunakan bahasa Python dan mengimplementasikan aspek-aspek pemrograman sebagai berikut.

A. Sekuens

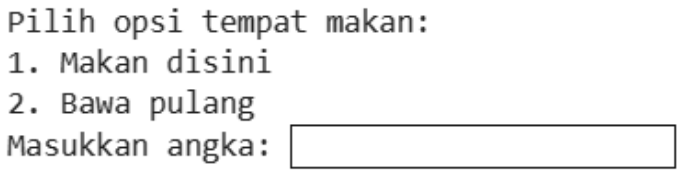
Dalam menyusun program ini, tentunya kami menggunakan sekuens agar program ini dapat berjalan secara prosedural. Kami mengurutkan tahapan-tahapan yang harus dilalui oleh pengguna untuk menggunakan simulasi mesin *self-ordering* ini dalam program dengan menggunakan sekuens.

B. Kondisional (Percabangan)

Dalam memesan makanan, tentunya akan ada banyak kemungkinan pilihan data yang dapat dimasukkan oleh pengguna. Variasi gabungan data juga sangat banyak. Oleh karena itu, penyusunan program ini pun tidak luput dari penggunaan percabangan. Beberapa bagian dari program mesin *self-ordering* ini yang menggunakan aspek percabangan adalah sebagai berikut.

1. Pemilihan opsi tempat makan

   Hal pertama yang harus dilakukan oleh pengguna setelah menjalankan program adalah memilih tempat makan. Pilihan yang disediakan adalah makan di tempat atau dibawa pulang. Pemilihan opsi ini tentunya akan menggunakan percabangan *(if, elif, else)* untuk menyimpan data yang dimasukkan oleh pengguna.



2. Pemilihan Kategori Menu

   Setelah memilih tempat makan, pengguna akan diminta untuk memilih kategori menu yang ingin dipesan. Dalam bagian ini program juga menggunakan percabangan, yakni ketika pengguna menginput masukan yang tidak sesuai dengan range list kategori menu, maka program akan membulatkannya ke angka terdekat.

```
Berikut kategori menu:
1. Promosi
2. Burger dan nuggets
3. Ayam
4. Happy meal
5. Paket keluarga
6. Menu receh
7. Camilan
8. Minuman
9. Pencuci mulut
10. Nasi
Silakan pilih kategori menu:
```

```
#kondisi kalau input < 1 atau > 10
if (masukan > 10):
  masukan = 10
elif (masukan < 1):
  masukan = 1
```

3. Konfirmasi Pemesanan

Selanjutnya, percabangan kondisi juga digunakan ketika program mengkonfirmasi penambahan pesanan. Pengguna dapat memilih untuk membatalkan pesanannya bila berubah pikiran. Jika pengguna membatalkan pesanannya, program akan menghapusnya dari pesanan. Sebaliknya, jika pengguna tidak ingin membatalkan pesanannya, program akan menambahkannya pada pesanan.

```
Apakah Anda ingin membatalkan pesanan ini? (y/n):
```

```
if (batal == 'y' or batal == 'Y'):
  # del arr_keranjang[i][0:3]
  arr_keranjang [i][0:3] = " "
else:
  sum_global += float(arr_keranjang [i][2])
```

4. Konfirmasi Penambahan Pesanan Lain

Setelah mengkonfirmasi penambahan menu makanan pada pesanan, program juga akan menanyakan apakah pengguna ingin menambahkan menu lain ke dalam pesanannya. Tentunya pada bagian ini juga akan menggunakan percabangan. Ketika pengguna ingin menambahkan pesanan, boolean pada program akan bernilai *True*. Sebaliknya, jika pengguna tidak ingin memesan menu lain, boolean pada program akan bernilai *False*.

```
Apakah Anda ingin menambahkan pesanan lain? (y/n):
```

```
if (konfirmasi == 'n' or konfirmasi == 'N'):
    boolpesan = False
else:
    i += 1
```

5. Jumlah Pesanan Pelanggan

Dalam menjalankan program ini, ada kalanya pelanggan tidak jadi memesan apapun. Maka, ketika hal tersebut terjadi, program tidak

akan melanjutkan ke tahapan selanjutnya dan menyampaikan bahwa pelanggan tidak memesan apapun.

```
#ketika tidak memesan apapun
if (i == 0 and (batal == 'y' or batal == 'Y')):
  print(" ")
  print("Anda tidak memesan item")

#ketika memesan 1 atau lebih item
else:
  #tempat ambil makan
  ambil_makan = 0
  str_ambil_makan = " "
```

6. Pemilihan Cara Pengambilan Makanan

Tahapan terakhir dari program ini setelah memasukkan menu-menu yang ingin dipesan oleh pengguna adalah memilih cara mengambil makanan. Pilihan yang disediakan adalah diambil sendiri ke konter atau diantarkan ke meja. Jika pengguna memilih untuk mengambil sendiri ke konter, maka program akan memintanya untuk memasukkan nomor pesanan. Jika pengguna ingin pesanannya diantar ke meja, maka pengguna akan diminta untuk memasukkan nomor meja.

```
Pilih opsi pengambilan makan:
1. Diantar ke meja
2. Diambil ke konter
Input angka:
```

```
if ambil_makan == 1:
    str_ambil_makan = "Diantar ke meja"
    no = int(input('Masukkan nomor meja: '))
elif ambil_makan == 2:
    str_ambil_makan = "Diambil ke konter"
    no = int(input('Masukkan nomor pesanan: '))
```

C. Loop

Looping digunakan dalam pemrograman untuk menjalankan ulang satu atau beberapa blok program dengan kondisi atau syarat tertentu yang harus dipenuhi. Dalam program simulasi mesin *self-ordering* ini, kami menggunakan aspek *looping* (pengulangan) pada bagian-bagian di bawah ini.

1. Pengulangan Input

Ketika pengguna memasukkan angka yang tidak sesuai dengan *range* yang disediakan program, program akan meminta pengguna untuk menginput ulang angka tersebut hingga mendapatkan angka yang sesuai dengan range.

```
  while ambil_makan < 1 or ambil_makan > 2:
      ambil_makan = int(input("""
Pilih opsi pengambilan makan:
1. Diantar ke meja
2. Diambil ke konter
Input angka:
"""))
```

```
#kondisi batal pesan atau tidak
batal = str(input("Apakah Anda ingin membatalkan pesanan ini? (y/n): "))
while (batal != "y" and batal != "Y" and batal != "N" and batal != "n"):
  batal = str(input("Apakah Anda ingin membatalkan pesanan ini? (y/n): "))
```

```
#kondisi pesan lagi atau tidak
konfirmasi = str(input("Apakah Anda ingin menambahkan pesanan lain? (y/n): "))
while (konfirmasi != "y" and konfirmasi != "Y" and konfirmasi != "N" and konfirmasi != "n"):
  konfirmasi = str(input("Apakah Anda ingin menambahkan pesanan lain? (y/n): "))
```

2. Pengulangan Proses Pemesanan

Pada program ini, pengguna dapat menambahkan lebih dari satu menu makanan pada pesanannya. Proses pemesanan per menu tentunya akan sama. Jadi, digunakan sistem pengulangan dalam bagian program ini untuk mengulang proses pemesanan. Nilai awal boolean yang digunakan untuk pengulangan ini adalah *True.* Ketika boolean terus bernilai *True*, maka program pemesanan akan berjalan secara berulang. Itulah sebabnya ketika program meminta konfirmasi penambahan pesanan kepada *user,* jika *user* tidak ingin menambahkan pesanannya lagi, program akan menyatakan nilai boolean sebagai *False* untuk menghentikan looping pada program pemesanan.

```
while (boolpesan == True):
```

D. *Function/Procedure*

Dalam penyusunan program simulasi mesin *self-ordering* ini, kami juga menggunakan beberapa fungsi dan juga prosedur sebagai bentuk subprogram untuk mempersingkat main program. Fungsi dan prosedur yang telah kami buat dalam program simulasi ini adalah sebagai berikut.

1. *Count*

Fungsi *count* dalam program ini berfungsi untuk meminta jumlah menu yang ingin dipesan, kemudian menghitung total harga satu menu makanan tersebut dan menambahkan data harga dari pesanan tersebut ke dalam "keranjang".

```
def count(harga):
    N = int(input("Berapa jumlah yang ingin dipesan? "))
    arr_keranjang [i][1] = str(N)
    arr_keranjang [i][2] = str(N * harga)
    return N*harga
```

2. *Process*

Prosedur *process* dalam program ini akan menampilkan seluruh menu yang ada dalam kategori menu pilihan pengguna. Kemudian, prosedur ini akan meminta pengguna untuk memasukkan angka menu yang ingin dipesan dan mengulangnya selama input dari pengguna tidak sesuai dengan *range* pilihan menu. Input ini kemudian akan digunakan untuk menambahkan menu yang dipesan tersebut ke dalam *list* keranjang. Setelahnya, prosedur *process* ini akan memanggil fungsi *count* untuk menghitung dan menambahkan harga pada keranjang pemesanan.

```
def process(x,len):

    #print out semua menu di kategori
    for loop in range(len):
        print(str(loop+1) + ". " + x[loop][0] + " (" + str(x[loop][1]) + "k)")
    N = int(input("Masukkan menu yang Anda inginkan: "))

    #input menu yang diinginkan
    while(N < 1 or N > len):
        N = int(input("Masukkan menu yang Anda inginkan: "))

    #proses barang ke struk
    arr_keranjang[i][0] = x[N-1][0]

    #proses harga
    num = (x[N-1][1])
    count(num)
    return
```

3. *Binserfirst & Binserlast*

Fungsi *binserfirst* dan *binserlast* dalam program ini adalah fungsi yang digunakan untuk mencari angka pertama dan terakhir dari data pada *database* yang telah ditetapkan. Dengan kata lain, fungsi ini akan mencara kapan mulai dan selesainya suatu kategori dalam *database. Database* akan di-*import* dari file yang berbeda.

```
#binary search angka pertama muncul
def binserfirst(low, high, key):
    index = 1
    while low <= high:
        mid = int(low + ((high-low)/2))
        if data['category'][mid] < key:
            low = mid + 1
        elif data['category'][mid] > key:
            high = mid - 1
        elif data['category'][mid] == key:
            index = mid
            high = mid - 1
    return index
```

```
#binary search angka terakhir muncul
def binserlast(low, high, key):
    index = 1
    while low <= high:
        mid = int(low + ((high-low)/2))
        if data['category'][mid] < key:
            low = mid + 1
        elif data['category'][mid] > key:
            high = mid - 1
        elif data['category'][mid] == key:
            index = mid
            low = mid + 1
    return index
```

E. Array & Matriks

Penyusunan program ini juga melibatkan penggunaan array dan matriks. Array dan matriks biasanya digunakan untuk menyajikan data dalam satu deretan atau lebih. Pembuatan program simulasi mesin *self-ordering* juga tentunya akan melibatkan penggunaan aspek array dan matriks. Beberapa bagian dari program simulasi kami yang menggunakan aspek array dan matriks tersebut adalah sebagai berikut.

1. Keranjang

Keranjang dalam program ini berfungsi sebagai matriks yang menyimpan menu-menu dan harga makanan yang telah dipesan oleh pengguna. Isi dari keranjang ini kemudian akan ditampilkan sebagai output dari program dalam bentuk struk.

```
out_data = arr_keranjang[0:i+1]
```

2. *Temporary*

Dalam program ini, *temporary* berfungsi sebagai matriks yang menyimpan data sementara pesanan pelanggan saat ini untuk kemudian dijalankan ke prosedur *process*.

```
#input data sementara ke array temporary
increase = 0
while(start <= finish):
    arr_temporary[increase][0] = data['item'][start]
    arr_temporary[increase][1] = data['price'][start]
    increase += 1
    start += 1

#proses
process(arr_temporary, increase)
```

3. *Data*

Data pada program ini juga berfungsi sebagai variabel matriks array yang berisi list menu dari *database*, mulai dari kategori, harga, dan nama menu.

```
# contoh list:
# kategori, price, item
# 1,13.2,Double Choco Sundae
# 1,39.2,3x Double Choco Pie
arr_temporary[increase][0] = data['item'][start]
arr_temporary[increase][1] = data['price'][start]
```

BAB IV
KESIMPULAN

A. Kesimpulan

Penyusunan program simulasi mesin *self-ordering* memfokuskan pada bagian pemesanan menu yang melibatkan penggunaan banyak aspek pemrograman, seperti sekuens, percabangan, pengulangan, fungsi dan prosedur, serta array dan matriks. Program yang kami buat hanya sebagian dari keseluruhan program yang digunakan dalam mesin *self-ordering*. Tentu pada realitanya, program pada *self-ordering machine* lebih kompleks dengan pertimbangan berbagai skenario yang mungkin tidak terpikirkan oleh kami. Meskipun demikian, tugas besar pengenalan komputasi ini telah menjadi sarana yang baik bagi kami untuk melatih kemampuan *programming*.

B. *Lesson Learned*

Banyak hal yang telah kami pelajari dalam proses pengerjaan tugas besar Pengenalan Komputasi ini, baik secara langsung maupun tidak langsung. Secara langsung, kami telah mengasah kemampuan *computational thinking*, melatih keterampilan menyusun *flowchart*, dan mengembangkan kemampuan *programming* dalam Python dengan mempelajari cara melibatkan berbagai aspek komputasi seperti sekuens, percabangan, pengulangan, fungsi dan prosedur, serta array dan matriks dalam program simulasi ini. Secara tidak langsung, dalam proses pengerjaan tugas besar ini, kami telah mempelajari cara untuk bekerja sama dan berkomunikasi dengan baik antar anggota kelompok.

PEMBAGIAN TUGAS

Dalam pengerjaan tugas besar Pengenalan Komputasi ini, pembagian tugas kelompok 7 adalah sebagai berikut.

1. Penyusunan Program	Nathan Jovial Hartono
2. Penyusunan Laporan	Jessica Allen
3. Penyusunan Flowchart	Naomi Risaka Sitorus
4. Penyusunan PPT	Angelina Efrina Prahastaputri

Walaupun pembagian tugas terdefinisi dengan jelas, setiap anggota tetap memberikan bantuan, tanggapan, perbaikan, serta saran mereka terhadap setiap bagian tugas.

SUMBER

Anonym. 2019. *Manfaat Teknologi Self-Order Untuk Bisnis Restoran Anda*. Surabaya. Interactive. https://interactive.co.id/blog/manfaat-teknologi-self-order-untuk-bisnis-restoran-anda-177.html#:~:text=Dengan%20sistem%20self%20order%2C%20pelanggan,campur%20tangan%20kasir%2F%20pelayan).

Salsabiela, Dzikrina. 2020. *Sistem Pemesanan Makanan Berkonsep* Self-Ordering *dan Berbasis Web pada Rumah Makan*. Surakarta. Universitas Muhammadiyah Surakarta. https://eprints.ums.ac.id/80889/1/naskah_publikasi_L200160091%283%29.pdf.

LAMPIRAN

PPT:

https://www.canva.com/design/DAFynpeZITE/l5Aap3b_XbBQRXJDqOOegw/edit?utm_content=DAFynpeZITE&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Video:

https://youtu.be/VAvc-OeiAkA?si=m1huoELBe7CsvvSs