**SWJR CAR RENTAL : PERSEWAAN MOBIL**

LAPORAN PROYEK AKHIR

MATA KULIAH COMP6362 – DATA STRUCTURE

KELAS BD20



Oleh:

2301939033 ALDIYAN MOES

2301953100 YODHATAMA GIRI

2301940861 KEVIN REYNALDI

Semester Genap (2019)

MALANG

**LEMBAR PERSETUJUAN PROYEK AKHIR**

**SWJR CAR RENTAL : PERSEWAAN MOBIL**

MATA KULIAH COMP6362 – DATA STRUCTURE

KELAS BD20

Semester Genap (2019)

Laporan akhir ini adalah benar karya kami :

**(ALDIYAN MOES) (YODHATAMA GIRI)**

**2301939033 2301953100**

**(KEVIN REYNALDI)**

**2301940861**

**Malang,.………………………**

**(Hanugra Aulia Sidharta)**

**D5850**

BAB I

LATAR BELAKANG

Struktur data merupakan salah satu unsur penting dalam merancang suatu program, khususnya program yang membutuhkan data dalam jumlah besar. Penggunaan struktur data akan mempermudah program dalam menyimpan dan mengambil data yang akan dimasukkan ke dalam sebuah program. Walau terlihat rumit dalam perancangannya, struktrur data akan sangat memudahkan bila program telah dijalankan. Sebagian besar developer menggunakan struktur data dalam penyusunan datanya, dikarenakan struktur data merupakan cara yang efisien untuk menyimpan data.

Dalam proyek ini kelompok kami berencana untuk merancang suatu program yang menggunakan konsep struktur data menggunakan bahasa C. Jenis struktur data yang akan kami gunakan adalah binary search tree. Kami menggunakan binary search tree karena perancangannya yang tidak serumit graph, namun tetap memiliki efiesiensi dalam mencari data, dibandingkan binary tree biasa. Konsep binary search tree ini akan kami aplikasikan pada program persewaan mobil, di mana pengguna dapat menambah data, mengubah data, mencari data, dan menghapus data. Selain itu, tujuan pembuatan program ini adalah untuk memenuhi tugas proyek akhir yang diberikan kepada kami.

BAB II

KAJIAN PUSTAKA

Struktur data adalah sebuah susunan khusus untuk mengatur, memroses, mengambil dan menyimpan data. Jenis struktur data apapun didesain untuk mengatur data dengan tujuan spesifik agar data dapat diakses dan bekerja dengan sesuai. Dalam pemrograman komputer, struktur data dapat digunakan atau dirancang agar dapat bekerja dengan berbagai jenis algoritma. Dalam tiap struktur data terdapat informasi tentang nilai data, hubungan antar data dan fungsi yang dapat diaplikasikan terhadap data tersebut. Struktur data memiliki penggunaan yang luas dan bermacam-macam dalam ruang lingkup ilmu komputer dan software engineering. Struktur data digunakan dalam hampir semua program atau sistem software yang telah dikembangkan. Selain itu, struktur data merupakan hal dasar dalam ilmu komputer dan software engineering. Sebagai seorang developer, mereka harus memiliki pengetahuan tentang struktur data yang baik.

Struktur data memiliki beberapa karakteristik. Beberapa karakteristik tersebut diantaranya, linear dan non-linear. Karakteristik ini menjelaskan apakah data disusun berdasarkan urutan, seperti array, atau berdasarkan urutan acak, seperti graph. Kemudian homogen dan non-homogen, di mana karakteristik ini menjelaskan apakah data yang ada memiliki tipe yang sama atau berbeda. Statis atau dinamis, karakteristik ini menjelaskan bagaimana struktur data dicompile. Struktur data statis memiliki ukuran yang tetap kapanpun. Struktur data dinamis memiliki ukuran yang dapat mengecil atau membesar berdasarkan penggunaannya.

Jenis struktur data ditentukan berdasarkan tipe operasi yang dibutuhkan atau algoritma yang akan digunakan. Berikut adalah beberapa jenis struktur data, array. Array menyimpan data dengan lokasi antar data yang saling berdampingan. Data dengan tipe yang sama disimpan dalam posisi di mana tiap elemen dapat dihitung atau diambil secara mudah. Panjang array dapat bersifat tetap atau fleksibel. Stacks, di mana dalam stack data disimpan dalam urutan linear dan memiliki urutan last in first out. Queue, memiliki prinsip yang sama dengan stack, namun memiliki urutan first in first out. Linked lists, tipe ini menyimpan data dalam urutan linear. Tiap elemen atau node dalam linked list memiliki data yang tersambung dengan data selanjutnya di dalam list. Trees, sebuah tree menyimpan data dengan susunan hirarki. Tiap node terhubung ke node lainnya dan bisa memiliki banyak sub-nilai, yang juga dikenal sebagai anak. Graphs, menyimpan data secara non-linear. Graphs terbuat dari kumpulan node yang juga dikenal sebagai vertices, dan garis yang menghubungkannya yang juga dikenal sebagai edges. Ini semua adaalah struktur data kompleks karena dapat menyimpan data dengan jumlah banyak. Contoh dari struktrur data dasar adalah integers, floats, booleans, dan characters.

Struktur data sangatlah penting untuk mengatur data dengan jumlah besar, seperti informasi yang disimpan dalam database secara efisien. Perawatan yang baik dari sistem data membutuhkan identifikasi dari alokasi memori, hubungan antar data, dan proses data, di mana struktur data dapat sangat memudahkan hal ini. Selain itu, tidak hanya penggunaan struktur data yang penting, tapi pemilihan struktur data yang tepat untuk tiap tugas juga penting. Beeberapa faktor untuk diperhatikan saat memilih struktur data adalah jenis informasi apa yang akan disimpan, di mana data akan ditempatkan, dan bagaimana data tersebut disortir dan berapa banyak memori yang harus dialokasikan untuk data.

Salah satu jenis struktur data adalah binary search tree. Binary search tree adalah struktur data yang memungkinkan kita untuk menyimpan angka. Disebut binary tree karena tiap node memiliki anak maksimal 2 cabang. Disebut search tree karena binary search tree dapat digunakan untuk mencari keberadaan sebuah angka dengan waktu 0(log(n). Beberapa hal yang membedakan binary search tree dengan binary tree yaitu, semua node di subtree sebelah kiri memiliki nilai yang lebih kecil daripada akar, dan semua node di subtree kanan memiliki nilai yant lebih besar daripada akar.

BAB III

GAMBARAN UMUM PROGRAM

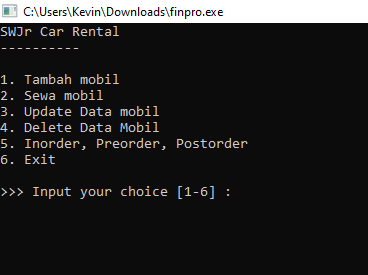
Kelompok kami membuat program untuk persewaan mobil drngan konsep binary search tree, di mana plat nomer yang telah diinput dapat dicari dan diubah datanya menggunakan konsep binary search tree. Selain dapat mencari dan mengubah data, program yang kami rancang dapat digunakan untuk menginput mobil atau item baru. Penentuan besar kecilnya nilai dalam tree dilakukan berdasarkan plat nomer mobil yang telah diinput.

Dalam program kami, pengguna dapat memasukkan mobil dan plat nomer baru, mengubah status mobil antara disewa atau tidak, mengupdate data mobil, menghapus data mobil, dan menampilkan data yang ada dengan urutan inorder, preorder, dan postorder. Untuk menambah data, pengguna harus memasukkan plat nomer, merek dan jenis mobil, harga sewa, dan status mobil tersebut apaakah sedang dipinjam atau tidak. Pada bagian menyewa mobil, pengguna harus memasukkan plat nomer mobil yang akan disewa. Ini berarti data mobil harus sudah ada, di mana dapat dilakukan dengan menambah data mobil. Setelah memasukkan plat nomer, program akan mencari plat nomer yang diinput, dan setelah ditemukan pengguna dapat mengubah statusnya apakah telah dipinjam atau tidak. Kemudian, pengguna dapat mengupdate data mobil dengan memasukkan plat nomer yang sudah ada. Update ini dilakukan untuk mengubah plat nomer bila plat nomernya diganti baru, ataupun bila ingin memberi perubahan harga. Pada bagian hapus data, pengguna dapat menghapus data dengan memasukkan plat nomer yang ada, dan program akan menghapus data mobil yang ada berdasarkan plat nomer yang sudah diinput. Hal ini dilakukan bila mungkin ada mobil yang telah dijual. Kemudian pengguna dapat memilih untuk melihat daftar mobil yang ada berdasarkan ururtan inorder, preorder, dan postorder. Kemudian pada pilihan akhir dapat digunakan untuk keluar dari program yang berjalan.

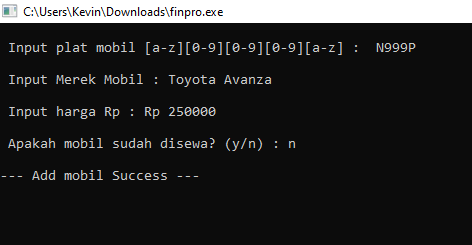
BAB IV

HASIL

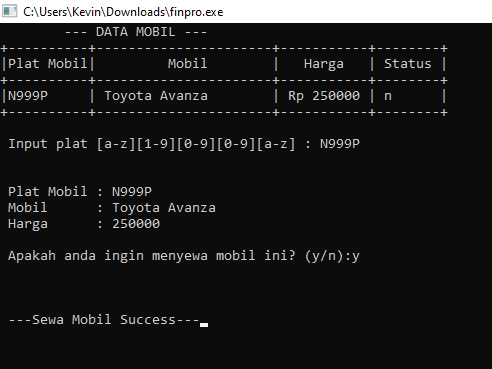
1. **Program Display**
2. Menu utama



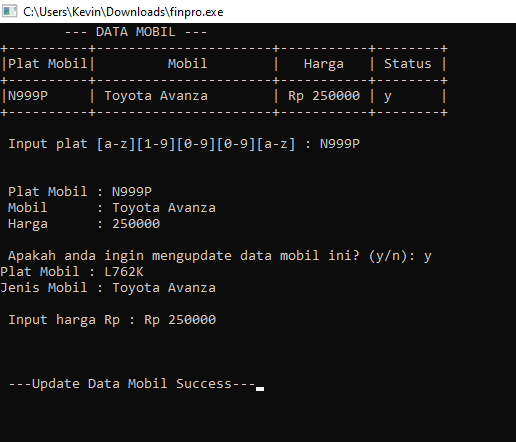
1. Tambah data mobil



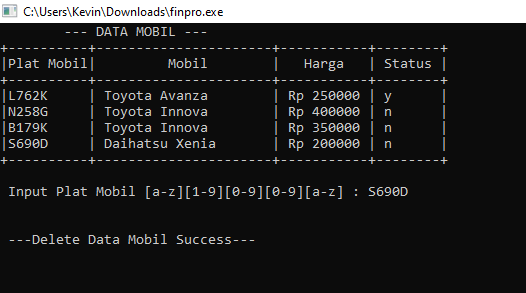
1. Sewa mobil



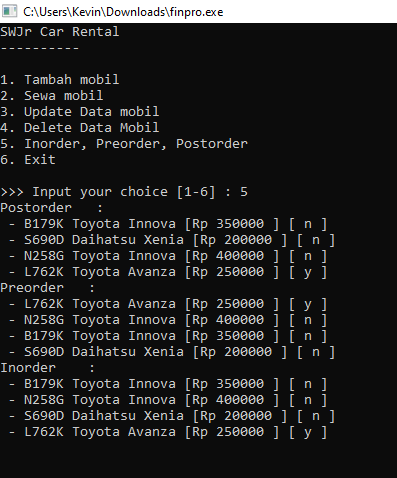
1. Update data mobil



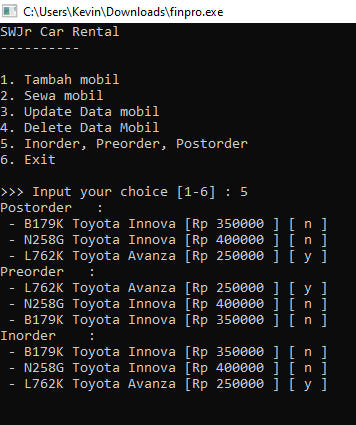
1. Hapus data mobil



1. Inorder, preorder, postorder



1. Tampilan setelah data dihapus



1. **Source Code**
2. #include<stdio.h>
3. #include<string.h>
4. #include<stdlib.h>
6. clear(){
7. system("cls");
8. }
10. **struct** node{
11. **char** plat[10];
12. **char** mobil[30];
13. **char** status[10];
14. **int** price;
15. **struct** node \*left;
16. **struct** node \*right;
17. };
18. **struct** node\* root = NULL;

21. **void** mainMenu(){
22. clear();
23. printf("SWJr Car Rental\n");
24. printf("----------\n\n");
26. printf("1. Tambah mobil\n");
27. printf("2. Sewa mobil\n");
28. printf("3. Update Data mobil\n");
29. printf("4. Delete Data Mobil\n");
30. printf("5. Inorder, Preorder, Postorder\n");
32. printf("6. Exit\n");
33. }

36. **struct** node \*newNode(**char** \*plat,**char** \*mobil,**char** \*status, **int** price){
37. **struct** node \*temp = (**struct** node\*)malloc(**sizeof**(**struct** node));
38. strcpy(temp->plat,plat);
39. strcpy(temp->mobil,mobil);
40. strcpy(temp->status,status);
41. temp->price = price;
42. temp->left = NULL;
43. temp->right = NULL;
44. };
46. **int** convert(**char** \*plat){
47. **char** src[10];
48. **int** key;
49. strcpy(src,plat);
50. memmove(&src[0],&src[1],strlen(src));//tuker n dapet kode angka
51. key = atoi(src);//convert to int
52. **return** key;
53. }

56. **void** tampil(**struct** node \*root){
57. **if**(root != NULL){
58. printf("|%-9s | %-20s | Rp %-3d | %-6s |\n",root->plat,root->mobil,root->price,root->status);
59. tampil(root->left);
60. tampil(root->right);
61. }
62. }
64. **struct** node\* insert(**struct** node \*temp, **char** \*plat,**char** \*mobil,**char** \*status, **int** price){
65. **int** key, node;
66. **if**(temp == NULL){
67. **return** newNode(plat,mobil,status,price);
68. }
69. key = convert(plat);
70. node = convert(temp->plat);
71. **if**(key < node){
72. temp->left = insert(temp->left,plat,mobil,status,price);
73. }
74. **else** **if**(key > node){
75. temp->right = insert(temp->right,plat,mobil,status,price);
76. }
77. **return** temp;
78. };
80. **struct** node \*findParent(**struct** node \*temp,**char** \*plat){
81. **if**(temp == NULL){
82. **return** NULL;
83. }
84. **if**(temp->left == NULL && temp->right == NULL){
85. **return** NULL;
86. }
87. **if**(temp->left != NULL && strcmp(temp->left->plat,plat) == 0 || temp->right != NULL && strcmp(temp->right->plat,plat) == 0){
88. **return** temp;
89. }
90. **int** key = convert(plat);
91. **int** keynode = convert(temp->plat);
92. **if**(keynode > key){
93. **return** findParent(temp->left,plat);
94. }
95. **if**(keynode < key){
96. **return** findParent(temp->right,plat);
97. }
98. }
100. **struct** node \*inorder(**struct** node \*temp){
101. **struct** node \*current = temp;
102. **while**(current && current->left != NULL){
103. current = current->left;
104. }
105. **return** current;
106. }
108. **struct** node \*del(**struct** node \*temp){
109. **if**(temp->left == NULL && temp->right == NULL){
110. **struct** node \*parent = findParent(root,temp->plat);
111. **if**(parent->left == temp){
112. parent->left = NULL;
113. }**else** **if**(parent->right == temp){
114. parent->right = NULL;
115. }
116. temp = NULL;
117. **return** temp;
118. }**else** **if**(temp->left == NULL){
119. **struct** node \*temp = temp->right;
120. strcpy(temp->plat,temp->plat);
121. strcpy(temp->status,temp->status);
122. strcpy(temp->mobil,temp->mobil);
123. temp->price = temp->price;
124. free(temp);
125. temp = NULL;
126. temp->right = temp;
127. **return** temp;
128. }**else** **if** (temp->right == NULL){
129. **struct** node \*temp = temp->left;
130. strcpy(temp->plat,temp->plat);
131. strcpy(temp->status,temp->status);
132. strcpy(temp->mobil,temp->mobil);
133. temp->price = temp->price;
134. free(temp);
135. temp = NULL;
136. temp->left = temp;
137. **return** temp;
138. }**else**{
139. **struct** node\* temp = inorder(temp->right);
140. strcpy(temp->plat,temp->plat);
141. strcpy(temp->status,temp->status);
142. strcpy(temp->mobil,temp->mobil);
143. temp->price = temp->price;
144. temp->right = del(temp->right);
145. }
146. **return** temp;
147. }
149. **struct** node\* search(**struct** node \*temp, **char** \*plat){
150. **int** key, node;
151. **if**(temp != NULL){
152. key = convert(plat);
153. node = convert(temp->plat);
154. **if**(node == key){
155. **return** temp;
156. }
157. **else** **if**(node < key){
158. **return** search(temp->right, plat);
159. }
160. **else**{
161. **return** search(temp->left, plat);
162. }
163. }
164. };
166. **void** addMobil(){
167. fflush(stdin);
168. clear();
169. **char** plat[10], mobil[30], status[2];
170. **int** price;
171. **do**{
172. printf("\n Input plat mobil [a-z][0-9][0-9][0-9][a-z] :  ");
173. gets(plat);
174. }**while**(strlen(plat) > 5);
175. **do**{
176. printf("\n Input Merek Mobil : ");
177. gets(mobil);
178. }**while**(strlen(mobil) < 1 || strcmp(mobil," ") == 0);
180. printf("\n Input harga Rp : Rp ");
181. scanf("%d",&price);

184. fflush(stdin);
185. **do**{
186. printf("\n Apakah mobil sudah disewa? (y/n) : ");
187. gets(status);
188. }**while**(strcmp(status,"y") != 0 && strcmp(status,"n") != 0);



193. **if**(root == NULL){
194. root = insert(root,plat,mobil,status,price);
195. printf("\n--- Add mobil Success ---");
196. }
197. **else** **if**(strcmp(plat,root->plat) == 0){
198. printf("\n\n --- Plat mobil sudah ada!! ---");
199. }
200. **else**{
201. insert(root,plat,mobil,status,price);
202. printf("\n--- Add mobil Success ---");
203. }
205. }
207. **struct** node \*sewaMobil(**struct** node \*temp){
208. **char** plat[10], status[2];
209. **int** found;
210. clear();
211. **if**(root == NULL){
212. printf("\n ---There is no data---");
213. **return**;
214. }
215. **else**{
216. printf("\t--- DATA MOBIL ---\n");
217. printf("+----------+----------------------+-----------+--------+\n");
218. printf("|Plat Mobil|         Mobil        |   Harga   | Status |\n");
219. printf("+----------+----------------------+-----------+--------+\n");
220. tampil(root);
221. printf("+----------+----------------------+-----------+--------+\n");
223. **do**{
224. printf("\n Input plat [a-z][1-9][0-9][0-9][a-z] : ");
225. fflush(stdin);
226. gets(plat);
227. }**while**(strlen(plat) > 5);
228. temp = search(temp,plat);
229. **if**(temp == NULL){
230. printf("\n\n ---Plat Mobil Tidak Ditemukan---");
231. getch();
232. **return** temp;
233. }
234. printf("\n\n Plat Mobil : %s",temp->plat);
235. printf("\n Mobil      : %s",temp->mobil);
236. printf("\n Harga      : %d",temp->price);
237. **do**{
238. printf("\n\n Apakah anda ingin menyewa mobil ini? (y/n):");
239. fflush(stdin);
240. gets(status);
241. }**while**(strcmp(status,"y") != 0 && strcmp(status,"n") != 0);
243. **if**(strcmp(status,"y") == 0){
244. strcpy(temp->status,status);
245. printf("\n\n\n ---Sewa Mobil Success---");
246. }
247. **else**{
248. printf("\n\n\n ---Sewa Mobil Dibatalkan---");
249. }
250. }
251. **return** temp;
252. }
254. **struct** node \*updateMobil(**struct** node \*temp){
255. **char** plat[10], mobil[30],choice;
256. **int** price;
257. clear();
258. **if**(root == NULL){
259. printf("\n ---There is no data---");
260. **return**;
261. }
262. **else** {
263. printf("\t--- DATA MOBIL ---\n");
264. printf("+----------+----------------------+-----------+--------+\n");
265. printf("|Plat Mobil|         Mobil        |   Harga   | Status |\n");
266. printf("+----------+----------------------+-----------+--------+\n");
267. tampil(root);
268. printf("+----------+----------------------+-----------+--------+\n");
270. **do**{
271. printf("\n Input plat [a-z][1-9][0-9][0-9][a-z] : ");
272. fflush(stdin);
273. gets(plat);
274. }**while**(strlen(plat) > 5);
275. temp = search(temp,plat);
276. **if**(temp == NULL){
277. printf("\n\n ---Plat Mobil Tidak Ditemukan---");
278. getch();
279. **return** temp;
280. }
281. printf("\n\n Plat Mobil : %s",temp->plat);
282. printf("\n Mobil      : %s",temp->mobil);
283. printf("\n Harga      : %d",temp->price);
284. **do**{
285. printf("\n\n Apakah anda ingin mengupdate data mobil ini? (y/n): ");
286. fflush(stdin);
287. scanf("%c",&choice);
288. }**while**(choice == 'y' && choice == 'n');
289. **if**(choice == 'y'){
290. **do**{
291. fflush(stdin);
292. printf("Plat Mobil : ");
293. gets(plat);
294. }**while**(strlen(plat) > 5);
295. **do**{
296. fflush(stdin);
297. printf("Jenis Mobil : ");
298. gets(mobil);
299. }**while**(strlen(mobil) < 1 || strcmp(mobil," ") == 0);
300. printf("\n Input harga Rp : Rp ");
301. scanf("%d",&price);
302. printf("\n\n\n ---Update Data Mobil Success---");
303. strcpy(temp->plat,plat);
304. strcpy(temp->mobil,mobil);
305. temp->price = price;
306. } **else**{
307. printf("\n\n ---Update dibatalkan---");
308. }
309. }
311. **return** temp;
312. }
314. **void** delMobil(**struct** node \*temp){
315. **if**(root == NULL){
316. printf("\n\n ---There is No Data---");
317. getch();
318. **return**;
319. }**else**{
320. **char** plat[10];
321. clear();
322. **if**(root == NULL){
323. printf("\n ---There is no data---");
324. getch();
325. **return**;
326. }
327. **else**{
328. printf("\t--- DATA MOBIL ---\n");
329. printf("+----------+----------------------+-----------+--------+\n");
330. printf("|Plat Mobil|         Mobil        |   Harga   | Status |\n");
331. printf("+----------+----------------------+-----------+--------+\n");
332. tampil(root);
333. printf("+----------+----------------------+-----------+--------+\n");
334. **do**{
335. printf("\n Input Plat Mobil [a-z][1-9][0-9][0-9][a-z] : ");
336. fflush(stdin);
337. gets(plat);
339. }**while**(strlen(plat) > 5);
340. temp = search(root,plat);
341. **if**(temp == NULL){
342. printf("\n\n ---Plat tidak ditemukan!---");
343. getch();
344. **return**;
345. }
346. **if**(temp == root && root->left == NULL && root->right == NULL){
347. root = NULL;
348. }**else**{
349. temp = del(temp);
350. }
351. printf("\n\n ---Delete Data Mobil Success---");
352. }
354. }
355. }
357. **void** post(**struct** node \*root){
358. **if**(root != NULL){
359. post(root->left);
360. post(root->right);
361. printf("\n - %s %s [Rp %2d ] [ %s ] ",root->plat,root->mobil,root->price,root->status);
362. }
363. }
365. **void** pre(**struct** node \*root){
366. **if**(root != NULL){
367. printf("\n - %s %s [Rp %2d ] [ %s ] ",root->plat,root->mobil,root->price,root->status);
368. pre(root->left);
369. pre(root->right);
370. }
371. }
373. **void** in(**struct** node \*root){
374. **if**(root != NULL){
375. in(root->left);
376. printf("\n - %s %s [Rp %2d ] [ %s ] ",root->plat,root->mobil,root->price,root->status);
377. in(root->right);
378. }
379. }
381. **void** transversal(){
382. **if**(root == NULL){
383. printf(" --- There is No data in The Tree ---");
384. }
385. **else**{
386. printf("Postorder   : ");
387. post(root);
388. printf("\nPreorder   : ");
389. pre(root);
390. printf("\nInorder    : ");
391. in(root);
392. }
393. }
395. **int** main(){
396. **int** pilih;
397. **do**{
398. **do**{
399. mainMenu();
400. printf("\n>>> Input your choice [1-6] : ");
401. scanf("%d",&pilih);
402. } **while**(pilih > 6 || pilih < 1);
404. **switch**(pilih){
405. **case** 1 : addMobil();
406. getch();
407. **break**;
408. **case** 2 : sewaMobil(root);
409. getch();
410. **break**;
411. **case** 3 : updateMobil(root);
412. getch();
413. **break**;
414. **case** 4 : delMobil(root);
415. getch();
416. **break**;
417. **case** 5 : transversal();
418. getch();
419. **break**;

422. }
423. } **while**(pilih < 6);
424. }

DAFTAR PUSTAKA

“Data Structures”.searchsqlserver.techtarget.com.10 Juni 2019.2 Juni 2020.https://l/definition/data-structure?amp=1

“8 Common Data Structures every Programmer Must Know”.towardsdatascience.com.28 Februari 2019.2 Juni 2020.https://towardsdatascience.com/8-common-data-structures-every-programmer-must-know-171acf6a1a42

“Binary Search Tree (BST).programiz.com.2 Juni 2020. https://www.programiz.com/dsa/binary-search-tree

LEMBAR PENILAIAN

**SWJR CAR RENTAL : PERSEWAAN MOBIL**

MATA KULIAH COMP6362 – DATA STRUCTURE

KELAS BD20

Semester Genap (2019)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DAFTAR MAHASISWA** | **NILAI** | | | | **BOBOT** | | | | **KREDIT** | | | | **TOTAL**  **KREDIT** |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| 2301939033 ALDIYAN MOES |  |  |  |  | 20% | 20% | 24% | 36% |  |  |  |  |  |
| 2301953100 YODHATAMA GIRI |  |  |  |  | 20% | 20% | 24% | 36% |  |  |  |  |  |
| 2301940861 KEVIN REYNALDI |  |  |  |  | 20% | 20% | 24% | 36% |  |  |  |  |  |
| **TOTAL** | | | | | | | | | | | | |  |

**KETERANGAN :**

* **Skala Penilaian : 0 sd. 100**
* **Komponen**

1 : Laporan

2 : Produk

3 : Pengetahuan

4 : Solusi

**Malang,…………………**

**HANUGRA AULIA SIDHARTA**

**D5850**