**ANALISIS DAN PENJELASAN PROGRAM JOBSHEET 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nomor Program** | **Baris Program** | **Petikan Source Code** | **Penjelasan** |
| 1 | 4 - 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; | Pendeklarasian tipe data bentukan bernama Node yang nantinya akan dihubungkan dengan Node lainnya melalui next dan prev sehingga membentuk double linked list |
| 1 | 11 - 12 | struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));new\_node->data = new\_data; | Pendeklarasian node baru bernama new\_node sekaligus mengalokasikan memori untuk node tersebut dilanjutkan dengan memasukkan data ke node tersebut |
| 1 | 13 - 14 | new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  if((\*head\_ref) != NULL){  (\*head\_ref)->prev = new\_node;}  (\*head\_ref) = new\_node; | Pointer next pada new\_node diarahkan kepada head dan pointer prev dari new\_node diarahkan kepada NULL, if statement menghubungkan prev dari head (jika ada) ke new\_node lalu mengubah new\_node menjadi head yang baru. |
| 1 | 23 - 34 | struct Node \*last;  printf("\nTraversal in forward direction\n");  while(node != NULL){  printf("%d ", node->data);  last = node;  node = node->next;  }  printf("\nTraversal in reverse direction\n");  while(last != NULL){  printf("%d ", last->data);  last = last->prev;  } | Menampilkan data dari node dimulai dari head sampai ujung dan Node last akan menampilkan data node dari ujung ke head. |
| 1 | 40 - 45 | struct Node \*head = NULL;  push(&head, 6);  push(&head, 5);  push(&head, 2);  printf("Created DLL is : ");  printList(head); | Inisialisasi linked list dengan NULL lalu beberapa tersebut fungsi push akan mengisi linked list menjadi NULL<-2-><-5-><-6->NULL |
| 2 | 4 – 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; struct Node \*new\_node = new Node();  // 2. Memasukkan data  new\_node->data = new\_data; | Pendeklarasian tipe data bentukan bernama Node yang nantinya akan dihubungkan dengan Node lainnya melalui next dan prev sehingga membentuk double linked list |
| 2 | 12-16 | struct Node \*new\_node = new Node();  new\_node->data = new\_data; if((\*head\_ref) == NULL){  (\*head\_ref) = new\_node;  } | Pengalokasian memori dan memberikan data untuk new\_node, new\_node akan menjadi head apabila linked list kosong |
| 2 | 19 - 24 | new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  // 4. ubah prev dari head menjadi new\_node  (\*head\_ref)->prev = new\_node;  // 5. pindahkan posisi head ke new\_node  (\*head\_ref) = new\_node; | Menghubungkan head dengan new\_node lalu menjadikan new\_node sebagai head baru |
| 2 | 39 - 47 | // 5. Ubah next dari prev\_node sebagai new\_node  prev\_node->next = new\_node;  // 6. Ubah prev dari new\_node sebagai prev\_node  new\_node->prev = prev\_node;  // 7. Ubah prev dari node next dari new\_node  if(new\_node->next != NULL){  new\_node->next->prev = new\_node;  } | Menghubungkan prev\_node dan new\_node menjadi prev\_node-><-new\_node-, jika sesudah new\_node masih terdapat node maka menjadi prev\_node-><-new\_node-><-new\_node.next |
| 2 | 51 - 61 | struct Node \*last;  printf("\nTraversal in forward direction\n");  while(node != NULL){  printf("%d ", node->data);  last = node;  node = node->next;  }  printf("\nTraversal in reverse direction\n");  while(last != NULL){  printf("%d ", last->data);  last = last->prev;  } | Menampilkan data yang terdapat pada node mulai dari node head sampai ke belakang dan sebaliknya menggunakan Node last |
| 3 | 4 - 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; struct Node \*new\_node = new Node();  // 2. Memasukkan data  new\_node->data = new\_data; | Pendeklarasian tipe data bentukan bernama Node yang nantinya akan dihubungkan dengan Node lainnya melalui next dan prev sehingga membentuk double linked list |
| 3 | 11 - 23 | Node \*new\_node = new Node();  //2. Masukkan data  new\_node->data = new\_data;  //3. Buat next dari new node sebagai head dan prev sebagai NULL  new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  //4. Ubah prev dari head node menjadi new\_node  if((\*head\_ref) != NULL){  (\*head\_ref)->prev = new\_node;  }  //5. Ubah posisi head ke new\_node  (\*head\_ref) = new\_node; | Mengalokasikan memori dan memasukkan data untuk new\_node, lalu menghubungkan new\_node dengan head (jika ada head) dan mengubah new\_node menjadi head baru |
| 3 | 28 - 33 | struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));  struct Node \*last = \*head\_ref; // digunakan pada step 5  //2. masukkan data  new\_node->data = new\_data;  //3. new\_node ini akan menjadi node terakhir, jadi ubah next nya ke NULL  new\_node->next = NULL; | Mendeklarasikan dan mealokasikan memori untuk Node new\_node dan Node last, Node last digunakan untuk menandai node terakhir. karena new\_node akan diletakkan di akhir node maka next dari new\_node di isi NULL. |
| 3 | 41 - 47 | while(last->next != NULL){  last = last->next;  }  //6. ubah next dari last node  last->next = new\_node;  //7. ubah prev dari new\_node menjadi last node  new\_node->prev = last; | Membuat new\_node menjadi node terakhir melalui last->next dan hubungkan new\_node dengan last |
| 3 | 52 - 63 | struct Node \*last;  printf("\nTraversal in forward direction\n");  while(node != NULL){  printf("%d ", node->data);  last = node;  node = node->next;  }  printf("\nTraversal in reverse direction\n");  while(last != NULL){  printf("%d ", last->data);  last = last->prev;  } | Menampilkan data yang terdapat pada node mulai dari node head sampai ke belakang dan sebaliknya menggunakan Node last |
| 4 | 4 - 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; | Pendeklarasian tipe data bentukan bernama Node yang nantinya akan dihubungkan dengan Node lainnya melalui next dan prev sehingga membentuk double linked list |
| 4 | 11 - 18 | struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));  new\_node->data = new\_data;  new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  if((\*head\_ref) != NULL){  (\*head\_ref)->prev = new\_node;  }  (\*head\_ref) = new\_node; | Mendeklarasikan dan mengalokasikan memori untuk new\_node dan menghubungkan new\_node dengan head (jika ada) lalu new\_node akan menjadi head yang baru |
| 4 | 29 - 37 | struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));  new\_node->data = new\_data;  new\_node->prev = next\_node->prev;  next\_node->prev = new\_node;  new\_node->next = next\_node; | Menghubungkan new\_node dengan next\_node menjadi new\_node-><-next\_node |
| 4 | 39 - 44 | if(new\_node->prev != NULL){  new\_node->prev->next = new\_node;  } //8. jika prev new\_node adalah NULL maka new\_node akan menjadi head  else{  (\*head\_ref) = new\_node; | Jika terdapat node sebelum new\_node maka menjadi new\_node.prev-><-new\_node-><-next\_node, jika tidak maka new\_node akan menjadi head yang baru |
| 4 | 48 - 59 | struct Node \*last;  printf("\nTraversal in forward direction\n");  while(node != NULL){  printf("%d ", node->data);  last = node;  node = node->next;  }  printf("\nTraversal in reverse direction\n");  while(last != NULL){  printf("%d ", last->data);  last = last->prev;  } | Menampilkan data yang terdapat pada node mulai dari node head sampai ke belakang dan sebaliknya menggunakan Node last |