Αναφορά Εργαστηρίου 2

Προεργασία

Ως προεργασία χρειάστηκε να δημιουργήσουμε ένα πρόγραμμα σε C το οποίο θα υλοποιούσε όλες τις λειτουργίες που μας είχαν ζητηθεί. Αφού ελέγξαμε αν έχει την επιθυμητή λειτουργικότητα τότε μετατρέψαμε μέρος του προγράμματος αυτού σε γλώσσα Clang.

Περιγραφή Ζητουμένων

Το ζητούμενο της άσκησης ήταν να υλοποιήσουμε μια συνδεδεμένη λίστα χρησιμοποιώντας μια δομή structure. Το πρόγραμμα έπρεπε να είχε και κάποιες λειτουργίες όπως εισαγώγη ενός κόμβου,διαγραφή και συγκεκριμένες εκτυπώσεις στοιχείων,διευθύνσεων και μεγεθών της λίστας.

Περιγραφή της Εκτέλεσης

Στην αρχή υλοποιήσαμε το πρόγραμμα σε C το οποίο κάλυπτε όλες τις συναρτήσεις που μας είχαν ζητηθεί. Έπειτα με προσοχή για να περάσουμε με ασφάλεια στην υλοποίηση σε γλώσσα Clang αντιστοιχήσαμε κάθε μεταβλητή του προγράμματος σε C με ένα καταχωρητή ίδιου τύπου. Μετατρέψαμε κάθε συνθήκη if..then..else σε συνθήκες με την εντολή goto και labels. Το εκτελέσιμο είχε τα αναμενόμενα αποτελέσματα.

Συμπεράσματα

Μετά το τέλος της άσκησης καταλάβαμε πιο καλά τι συμβαίνει με τις διευθύνσεις μνήμης και τις μεταβλητές. Δηλαδή τον χώρο αποθήκευσης και το μέγεθος που καταλαμβάνει για παράδειγμα ολόκληρη η λίστα ή ακόμα και ο κάθε κόμβος ξεχωριστά.

Παράρτημα - Κώδικες - Flowcharts

Κώδικας Προγράμματος σε C:

```
#include <stdio.h>
#include <stdlib.h>

struct list {
  short  value;
  int id;
  struct list *next;
};

int menuPrint();
struct list * createList();
struct list *insertNodeList(int i,short val, struct list **head);
struct list *deleteNodeList(struct list **head);
void printNode(struct list *head);
int printNumOfNodes(struct list *head);
void printAddressNode(struct list *head);
void printAddressList(struct list * head);
void printSpecialNode(struct list* head);
void printSizeList(int help);
void printSizeNode();
void printAll(struct list *head,int i);

int main(void){

    struct list *node = NULL;
    int choice;
    int num;
    int i,j=0,b;
    short v;

    do{
        choice = menuPrint();
        if(choice == 1){
                node = createList();
        }
        else if(choice == 2){
                printf("Give value\n");
        scanf("%d",&v);
        printf("Give ID\n");
        scanf("%d",&i);
        insertNodeList(i,v,&node);
                printAll(node,1);
        }
                else if(choice == 3){
                        deleteNodeList(&node);
                        printAll(node,1);
                }
```

```
                else if(choice == 4){
                        printNode(node);
                }
                else if(choice == 5){
                        b = printNumOfNodes(node);
                        printf("The number of nodes is %d\n",b);
                }
                else if(choice == 6){
                        printAddressNode(node);
                }
                else if(choice == 7){
                        printAddressList(node);
                }
                else if(choice == 8){
                        printSpecialNode(node);
                }
                else if(choice == 9){
                        b = printNumOfNodes(node);
                        printSizeList(b);
                }
                else if(choice == 10){
                        printSizeNode();
                }
                }while(choice != 11);


   return (EXIT_SUCCESS);
}

int menuPrint(){
        int ch;
        printf("\n\n\n\nCreate List (1)\n");
        printf("Insert a node (2)\n");
        printf("Delete the 1st node (3)\n");
        printf("Print special item (4)\n");
        printf("Print the number of items (5)\n");
        printf("Print special address of item (6)\n");
        printf("Print the address of list (7)\n");
        printf("Print the address of special item section (8)\n");
        printf("Print the size of list (9)\n");
        printf("Print the size of item (10)\n");
        printf("Exit (11)\n");
        scanf("%d",&ch);
        return ch;
}

struct list * createList(){
         struct list *node = NULL;
```

```
        return node;
}

struct list *insertNodeList(int i,short val, struct list **head){

   struct list *node;
   struct list **node2 = head;
   node = (struct list *)malloc(sizeof(struct list));
   node->value = val;
   node->id = i;
   node->next = NULL;

   while (*node2 != NULL){

                node2 = &((**node2).next);
   }

   *node2 = node;
        return *head;
}

struct list * deleteNodeList(struct list **head) {

  if (*head != NULL){
    *head = (*head)->next;
  }
  else{
        printf("Empty List!\n");
        return;
  }
  return *head;
}

void printNode(struct list *head){

        struct list* node = head;
        int s;
        printf("Give node\n");
        scanf("%d",&s);
        int i=0;
        while(i<s){
                if((i == (s-1)) && (node != NULL)){
                        printf("Node info:\n\nValue: %d\nID: %d\n",node->value,node->id);
                }
                i++;
                node = node->next;
        }
        return;
```

```
}

int printNumOfNodes(struct list *head){
	struct list* node = head;
	int i=0;
	while(node != NULL){
		i++;
		node = node->next;
	}
	return i;
}

void printAddressNode(struct list *head){

	struct list* node = head;
	int s;
	printf("Give node\n");
	scanf("%d",&s);
	int i=0;
	while(i<s){
		if((i == (s-1)) && (node != NULL)){
			printf("Node info:\n\nValue: %d\nID: %d\n",node->value,node->id);
			printf("Addresses:\nValue: %d\nID: %d\n",&node->value,&node->id);
			i++;
		}
		node = node->next;
	}
	return;
}

void printAddressList(struct list * head){
	struct list* node = head;
	printf("Address of list: %d\n",&head);
}

void printSpecialNode(struct list* head){
	struct list* node = head;
	int s1;
	char ch;
	short s2;

	printf("Value or ID?(v-i)\n");
	scanf("\n%c",&ch);
	if(ch == 'i'){
		printf("Give id\n");
		scanf("%d",&s1);
		while(node != NULL){
			if(node->id == s1){
```

```
                        printf("ID address: %d\n",&node->id);
                    }
                    node = node->next;
                }
        }
        else if (ch == 'v'){
                printf("Give value\n");
                scanf("%d",&s2);
                while(node != NULL){
                    if(node->value == s2){
                        printf("Value address: %d\n",&node->value);
                    }
                    node = node->next;
                }
        }
        return;
}

void printSizeList(int help){
        if(help == 0){
                printf("Size of list: %d\n",sizeof(struct list));
        }
        else{
        printf("Size of list: %d\n",help*sizeof(struct list));
        }
}

void printSizeNode(){
        printf("Size of list: %d\n",sizeof(struct list));
}

void printAll(struct list *head,int i){
        if(head == NULL){
                return;
        }
        else {
                printf("Node %d:\n",i);
                printf("Value: %d\nID: %d\n\n\n",head->value,head->id);
                printAll(head->next,i+1);
        }
        return;
}
```

Κώδικας Προγράμματος σε Clang:

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct list {
  short  value;
  int id;
  struct list *next;
};

int menuPrint();
struct list * createList();
struct list *insertNodeList(int i,short val, struct list **head);
struct list *deleteNodeList(struct list **head);
void printNode(struct list *head);
void printAll(struct list *head,int i);
int main(void){

   struct list *node = NULL;
   int choice;
   int num;
   int i,j=0,b;
   short v;

   struct list * R1;
    R1 = NULL;
    int R0;
    R0 = 0;
    int R2,R3,R4,R5,R6;
    R5 = 0;
    short R7;

    do_label:
          R2 = menuPrint();
          if(R2 != 1) goto else_label_1;
                    R1 = createList();
                    goto after_cond;
                    else_label_1:
                    if(R2 != 2) goto else_label_2;
                              printf("Give value\n");
          scanf("%d",&R7);
          v = R7;
          printf("Give ID\n");
          scanf("%d",&R4);
          i = R4;
          insertNodeList(R4,R7,&R1);
          printAll(R1,1);
          goto after_cond;
          else_label_2:
          if(R2 != 3) goto else_label_3;
                    deleteNodeList(&R1);
```

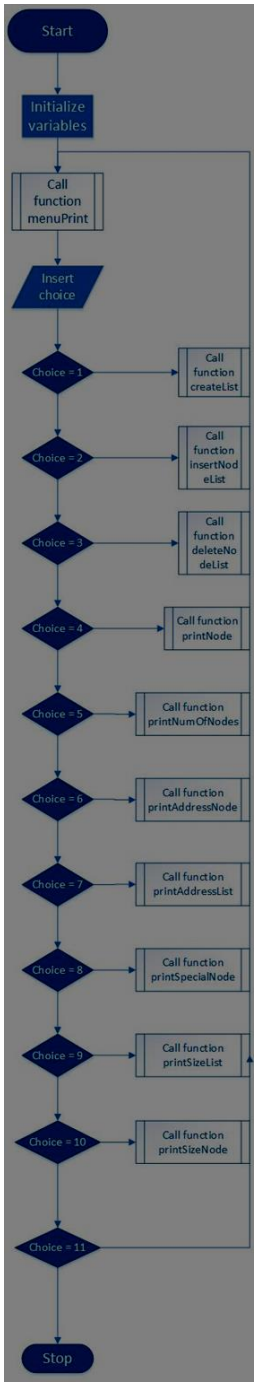
```
                printAll(R1,1);
        goto after_cond;
        else_label_3:
        if(R2 != 4) goto after_cond;
                printNode(R1);
        after_cond:
                if(R2 == 5) goto after_loop;
    goto do_label;
    after_loop:


    return (EXIT_SUCCESS);
}

int menuPrint(){
        int R8;
        int ch;
        printf("\n\n\nCreate List (1)\n");
        printf("Insert a node (2)\n");
        printf("Delete the 1st node (3)\n");
        printf("Print special item (4)\n");
        printf("Exit (5)\n");
        scanf("%d",&R8);
        ch = R8;
        return ch;
}

struct list * createList(){
         struct list *R1 = NULL;
         return R1;
}

struct list *insertNodeList(int R4,short R7, struct list **R1){
   struct list *node;
   struct list **node2;
   node2 = R1;
   struct list *R9,**R10;
   R9 = node;
   R10 = node2;


   R9 = (struct list *)malloc(sizeof(struct list));
   R9->value = R7;
   R9->id = R4;
   R9->next = NULL;

   while_label:
   if(!(*R10 != NULL)) goto after_loop;
```

```
        R10 = &((**R10).next);
   goto while_label;
   after_loop:

   *R10 = R9;

        return *R10;
}

struct list * deleteNodeList(struct list **R1) {

  if (*R1 == NULL) goto else_label;
    *R1= (*R1)->next;
  goto after_cond;
  else_label:
        printf("Empty List!\n");
        return;
  after_cond:
  return *R1;
}

void printNode(struct list *R1){

        struct list * R11;
        R11 = R1;
        int R10;
        int s;
        printf("Give id\n");
        scanf("%d",&R10);
        while_label:
        if(!(R11 != NULL)) goto after_loop;
                if(!(R11->id == R10)) goto after_cond;
                        printf("Node info:\n\nValue: %d\nID: %d\n",R11->value,R11->id);
                after_cond:
                R11 = R11->next;
        goto while_label;
        after_loop:
        s = R10;
        return;
}

void printAll(struct list *R1,int R12){
        struct list * R13;
        R13 = R1;
        if(!(R13 == NULL)) goto else_label;
                return;
        goto after_cond;
        else_label:
```

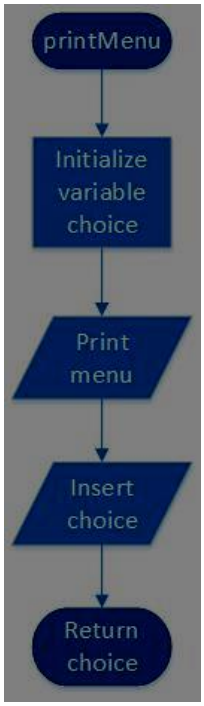
```
                printf("Node %d:\n",R12);
                printf("Value: %d\nID: %d\n\n\n",R13->value,R13->id);
                printAll(R13->next,R12+1);
        after_cond:
        return;
}
```

Flowcharts

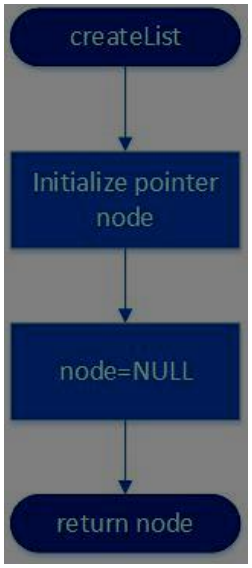
main:



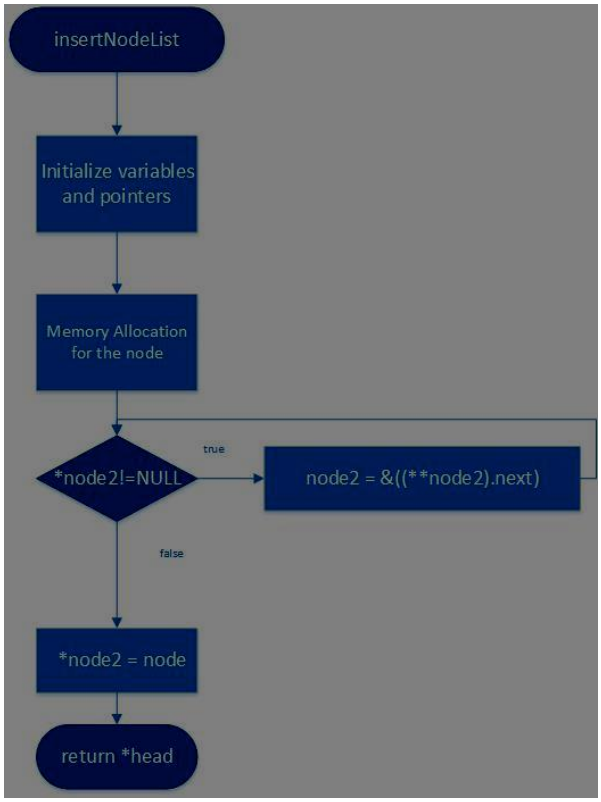
printMenu:



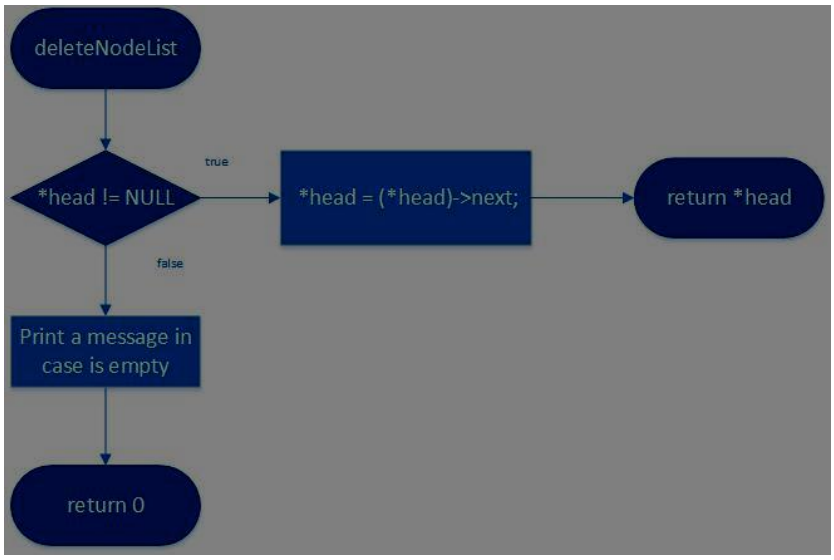
createList:



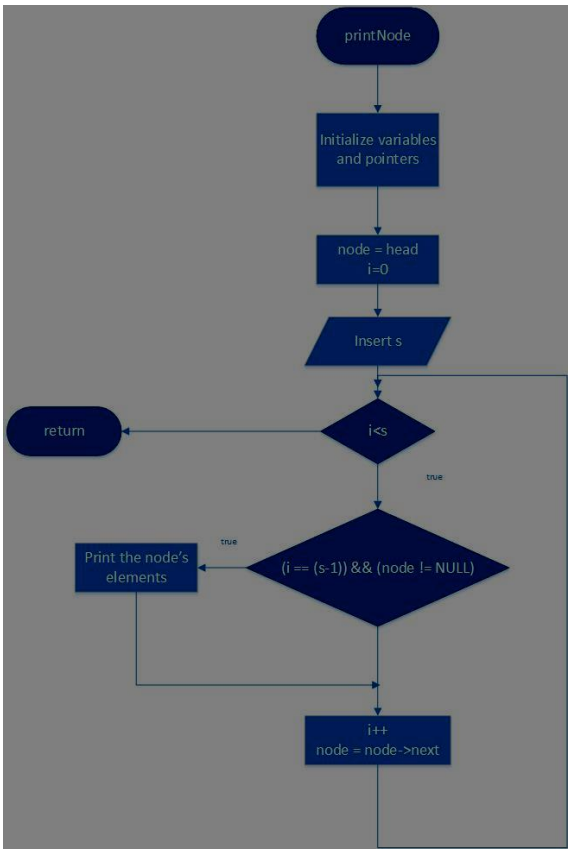
insertNodeList:



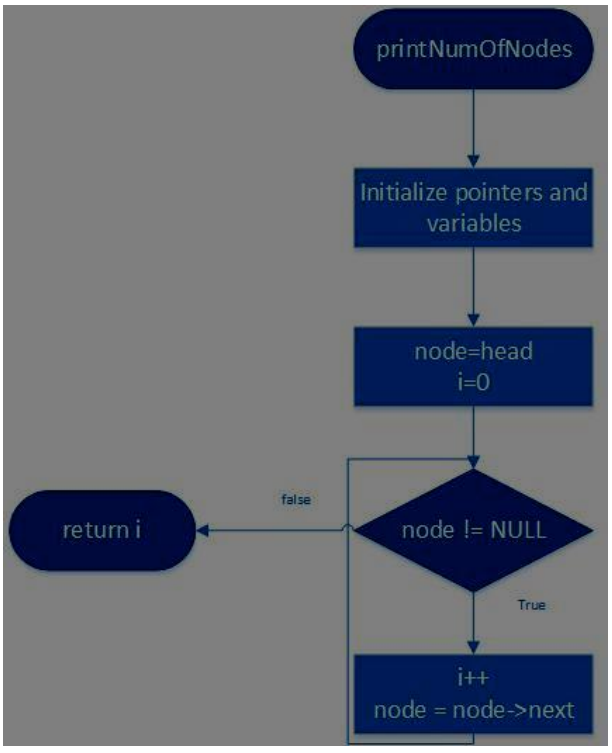
deleteNodeList:



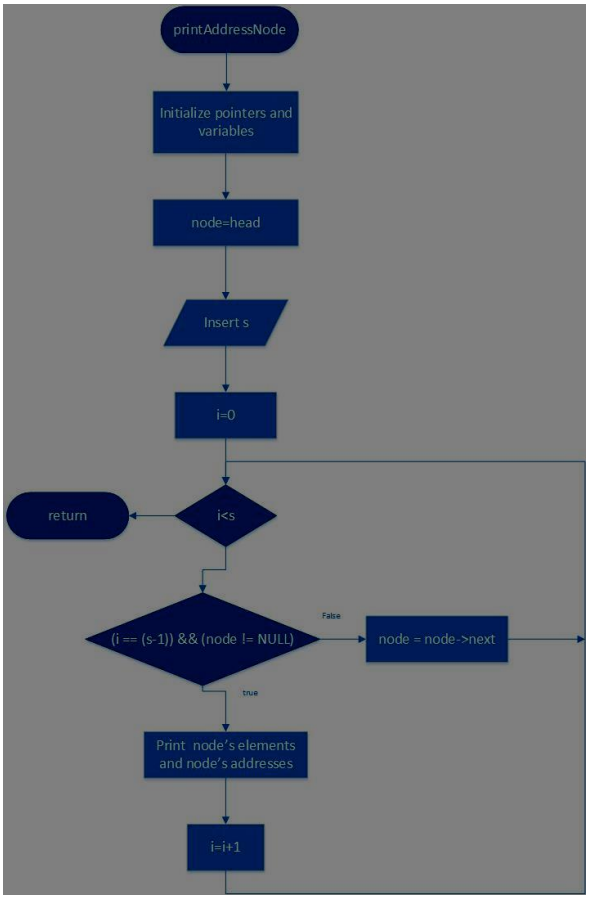
printNode:



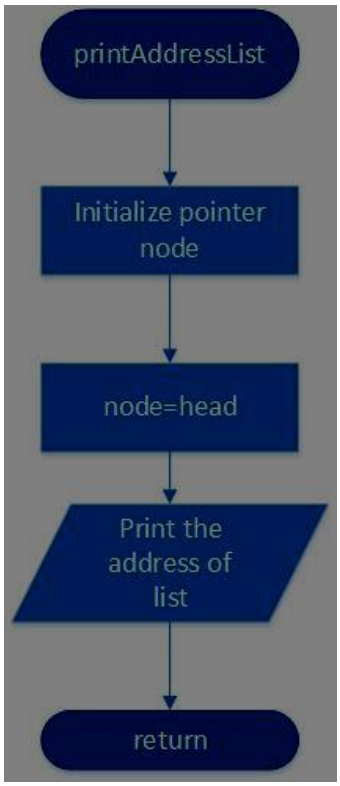
printNumOfNodes:



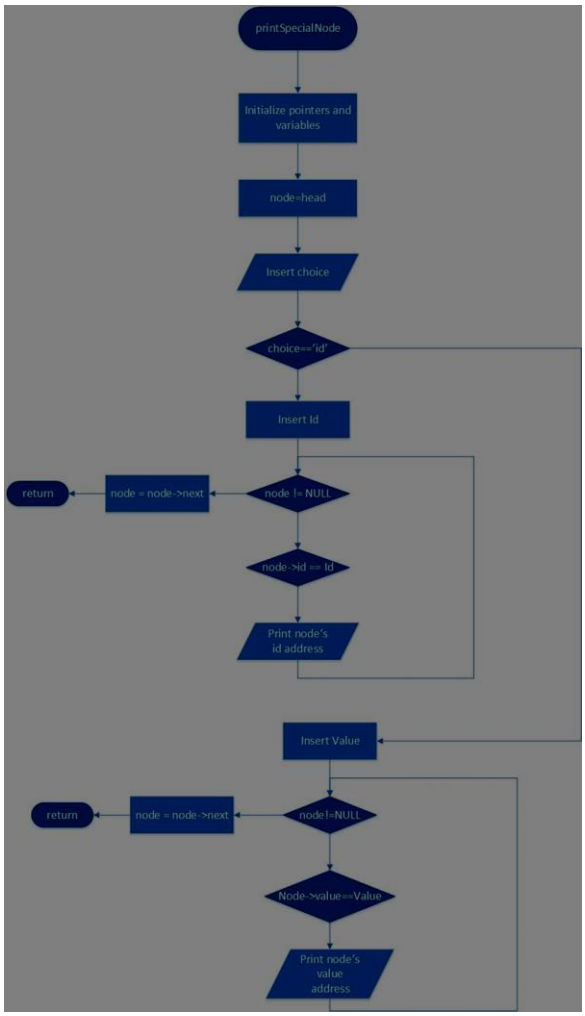
printAddressNode:



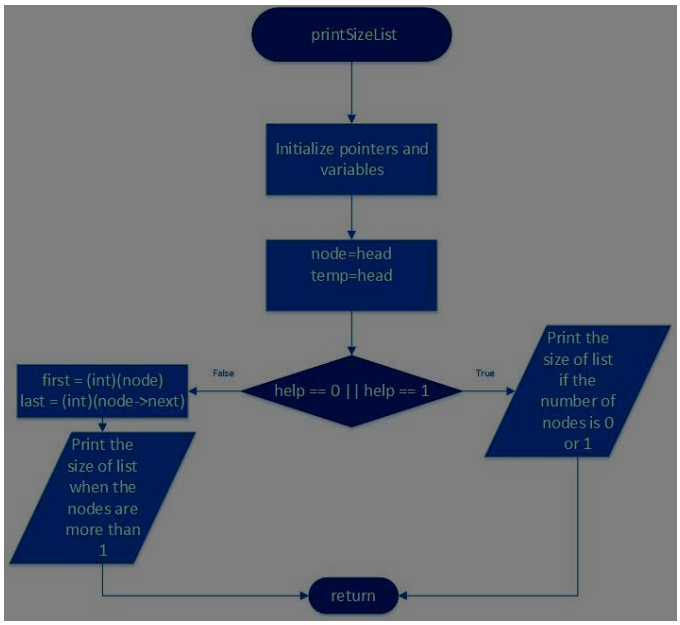
printAddressList:



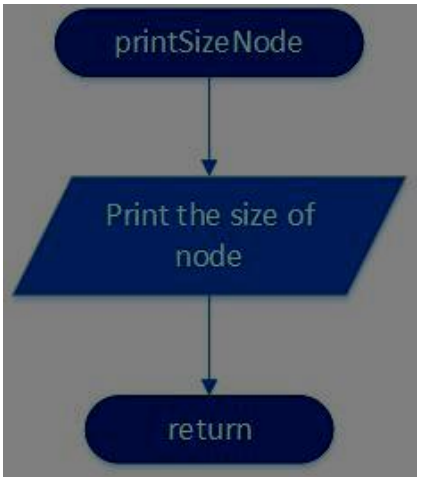
printSpecialNode:



printSizeList:



printSizeNode:


printSizeNode
Print the size of node
return