Lab 4:

Task 1:

1)

void insertNodeAtEnd(struct node\_d \*\* head)

{

struct node\_d \* temp = \*head;

///create a new node

struct node\_d \* new\_node = (struct node\_d\*) malloc(sizeof(struct node\_d));

inputNodeData(new\_node); /// get data for the newly created node from the user.

if(isListEmpty(\*head)) /// if currently the list is empty

{

\*head = new\_node;

///point its next and previous pointers to the start

new\_node->next = \* head;

new\_node->prev = \* head;

}

else

{

while((temp->next) != \*head)

temp = temp->next; /// scroll to the end of the list

temp->next = new\_node;

new\_node->next = \* head; /// Point to the first location

new\_node->prev = temp;

(\*head)->prev= new\_node; /// Point to the second last item

}

}

2)

int saveListToFile(struct node\_d \* head, FILE \* fptr)

{

struct node\_d \* temp = head;

if(head == NULL)

return(-1);

do

{

fwrite(&(temp->data), sizeof(struct employee), 1, fptr);

temp = temp->next; /// scroll to the end of the list

}

while(temp != head);

return(0);

}

Task 2:

a)

void deleteNodeFromStart(struct node\_d \* head)

{

if(isListEmpty(head))

{

printf("\n there's no node for deletion\n");

return;

}

if(head->next == head)

{

free(head);

head=NULL;

printf("\nNode Deleted\n");

}

else

{

struct node\_d \*temp= head;

struct node\_d \*tail= (head)->prev;

struct node\_d \*aft = (head)->next;

tail->next=aft;

head=aft;

(head)->prev=tail;

free(temp);

}

printf("\nNode Deleted\n");

}

b)

int deleteNodeAfter(struct node\_d \* head, int idx)

{

/\*\*\* To be completed by the students \*\*\*/

struct node\_d \*temp=head;

struct node\_d \*del=temp->next;

struct node\_d \*aft=del->next;

if(getListLength(head)==0 || idx ==0)

{

printf("\n there's no node for deletion\n");

return(-1);

}

for(int i=0; i<idx;i++)

temp=temp->next;

if(temp->next == head)

{

printf("\nthere's only two node, it cannot be deleted\n");

return(-1);

}

temp->next=aft;

aft->next=temp;

free(del);

return(0);

}

c)

int insertNodeAfter(struct node\_d \* head, int idx)

{

int index = 0;

struct node\_d \* temp = head;

if(isListEmpty(head)) /// if currently the list is empty return -1

{

return(-1);

}

///create a new node

struct node\_d \* new\_node = (struct node\_d \*) malloc(sizeof(struct node\_d));

inputNodeData(new\_node); /// get data for the newly created node from the user.

while((index != idx) && (temp->next) != head)

{

index ++;

temp = temp->next; /// scroll to the required index

}

if((temp->next == head) && (index != idx)) /// We reached the end of the list without

{ /// reaching the required index

free(new\_node);

printf("\nNode insertion not done. New data is discarded!!\n");

return(-1);

}

new\_node->next = temp->next; /// Setting up the pointers for insertion

new\_node->prev = temp;

temp->next = new\_node;

new\_node->next->prev = new\_node;

return(0); /// Successfully inserted node at idx

}

Post lab:

void printMemMap(struct node\_d \* head)

{

/\*\*\* To be completed by the students \*\*\*/

int x= getListLength(head);

struct node\_d \*temp= head;

printf("No \t Address \t Data \t Next \t Prev \n");

int count;

for(count=0; count<x;count++);

{

printf("%d\t%u\t%d\t%u\t%u\n",count,temp,temp->data.age,temp->next,temp->prev);

temp=temp->next;

}

}