```
#include<iostream>
#include<string>
#include<stdlib.h>                                          // clear screen

using namespace std;

struct Node {
public:
	int data;
	Node* nextNode;
};

class Linklist {
private:
	int size;
	Node* head;
	Node* current;
	Node* previous;
public:
	Linklist() {
		size = 0;
		head = NULL;
		current = NULL;
		previous = NULL;
	}
	int getSize() {
		return size;
	}

	//	Insertion	//
	void insertStart(int a) {
		Node* newNode = new Node();
		newNode->data = a;
		if (head == NULL) {
			head = newNode;
			newNode->nextNode = NULL;
```

```
        }
        else {
                newNode->nextNode = head;
                head = newNode;
        }
        size++;
}
void insertEnd(int a) {
        Node* newNode = new Node();
        newNode->data = a;
        if (head == NULL) {
                head = newNode;
                newNode->nextNode = NULL;
        }
        else {
                current = head;
                while (current->nextNode != NULL) {
                        current = current->nextNode;
                }
                current->nextNode = newNode;
                newNode->nextNode = NULL;
        }
        size++;
}
void insertSpecific(int a, int location) {
        Node* newNode = new Node();
        newNode->data = a;
        if (head == NULL && location == 1) {
                head = newNode;
                newNode->nextNode = NULL;
                size++;
        }
        else if (location <= size + 1) {
                if (location == 1) {
                        newNode->nextNode = head;
                        head = newNode;
```

```
		}
		else {
			previous = head;
			current = head;
			int i = 1;
			while (i != location) {
				previous = current;
				current = current->nextNode;
				i++;
			}
			previous->nextNode = newNode;
			newNode->nextNode = current;
		}
		size++;
	}
	else {
		cout << "Invalid Location\n";
	}
}



//	deletion	//
void deleteStart() {
	if (head == NULL)
		cout << "List is Empty";
	else {
		Node* temp;
		temp = head;
		head = head->nextNode;
		delete temp;
		size--;
	}
}
void deleteEnd() {
	if (head == NULL)
		cout << "List is Empty\n";
```

```
        else {
                if (size == 1) {
                        Node * temp = head;
                        head = NULL;
                        delete temp;
                        size--;
                }
                else {
                        Node* temp;
                        temp = head;
                        previous = head;
                        while (temp->nextNode != NULL) {
                                previous = temp;
                                temp = temp->nextNode;
                        }
                        previous->nextNode = NULL;
                        delete temp;
                        size--;
                }
        }
}
void deleteSpecific(int location) {
        if (head == NULL)
                cout << "List is Empty";
        else if (location <= size) {
                if (location == 1 && size == 1) {
                        Node* temp = head;
                        head = NULL;
                        delete temp;
                        size--;
                }
                else if (location == 1 && size > 1) {
                        Node* temp = head;
                        head = head->nextNode;
                        temp->nextNode = NULL;
                        delete temp;
```

```
                        size--;
                }
                else {
                        Node* temp = head;
                        previous = head;
                        int i = 1;
                        while (i != location) {
                                previous = temp;
                                temp = temp->nextNode;
                                i++;
                        }
                        previous->nextNode = temp->nextNode;
                        delete temp;
                        current = head;
                        size--;
                }
        }
        else {
                cout << "\n\t\tLocation is out if range\n";
        }
}



//      display//
void displayAll() {
        if (head == NULL)
                cout << "!!!--Empty List--!!!\n";
        else {
                current = head;
                while (current->nextNode != NULL) {
                        cout << current->data << " ";
                        current = current->nextNode;
                }
                cout << current->data << endl;
        }
}
```

```
void displayOdd() {
	if (head == NULL)
		cout << "!!!--Empty List--!!!\n";
	else {
		current = head;
		while (current->nextNode != NULL) {
			if (current->data % 2 == 1) {
				cout << current->data << " ";
				current = current->nextNode;
			}
			else
				current = current->nextNode;
		}
		if (current->data % 2 == 1) {
			cout << current->data << " \n";
		}
	}
}
void displayEven() {
	if (head == NULL)
		cout << "!!!--Empty List--!!!\n";
	else {
		current = head;
		while (current->nextNode != NULL) {
			if (current->data % 2 == 0) {
				cout << current->data << " ";
				current = current->nextNode;
			}
			else
				current = current->nextNode;
		}
		if (current->data % 2 == 0) {
			cout << current->data << " \n";
		}
	}
}
```

```
//      Swap    //
void swapValue(int a, int b) {
        if (head == NULL) {
                cout << "!!!--Empty List--!!!\n";
        }
        else {
                previous = head;
                current = head;

                while (previous->nextNode != NULL) {
                        if (previous->data == a) {
                                break;
                        }
                        else
                        previous = previous->nextNode;
                }
                while (current->nextNode != NULL) {
                        if (current->data == b) {
                                break;
                        }
                        else
                        current = current->nextNode;
                }

                if (previous->data == a && current->data == b) {
                        swap(previous->data, current->data);
                }
                else {
                        cout << "Entered value(s) doesnot found!\n";
                }
        }
}
void swapLocation(int loc1, int loc2) {
        if (head == NULL) {
```

```
		cout << "!!!--Empty List--!!!\n";
	}
	else {
		if (loc1 <= size && loc2 <= size) {
			previous = head;
			current = head;
			for (int i = 1; i < loc1; i++) {
				previous = previous->nextNode;
			}
			for (int j = 1; j < loc2; j++) {
				current = current->nextNode;
			}
			swap(previous->data, current->data);
		}
		else {
			cout << "Entered location(s) is/are out of range\n";
		}
	}
}



//	Copying//
void copyLoc(int loc1, int loc2) {
	if (head == NULL) {
		cout << "!!!--Empty List--!!!\n";
	}
	else {
		if (loc1 <= size && loc2 <= size) {
			previous = head;
			current = head;
			for (int i = 1; i < loc1; i++) {
				previous = previous->nextNode;
			}
			for (int j = 1; j < loc2; j++) {
				current = current->nextNode;
			}
```

```
                        int temp = previous->data;
                        current->data = temp;
                }
                else {
                        cout << "Entered location(s) is/are out of range\n";
                }
        }
}


// Update      //
void update(int val, int loc) {
        if (head == NULL) {
                cout << "!!!--Empty List--!!!\n";
        }
        else {
                if (loc <= size) {
                        current = head;
                        for (int i = 1; i < loc; i++) {
                                current = current->nextNode;
                        }
                        current->data = val;
                }
                else {
                        cout << "Entered location(s) is/are out of range\n";
                }
        }
}


// Find Values //
void findVal(int a) {
        if (head == NULL) {
                cout << "!!!--Empty List--!!!\n";
        }
        else {
```

```
                        current = head;
                        int count = 0;
                        while (current->nextNode != NULL) {
                                count++;
                                if (current->data == a) {
                                        break;
                                }
                                current = current->nextNode;
                        }
                        if (current->data == a) {
                                cout << "Your value : " << a << " is found at position : " << count
<< endl;
                        }
                        else {
                                cout << "Value not found\n";
                        }
                }
        }



        //      Length  //
        void len() {
                cout << "your linkist's length is : " << size << endl;
        }



        // delete List //
        void delList() {
                head = NULL;
                cout << "!!!---List deleted---!!!\n";
        }

};

void goAhead() {
        cin.ignore();
        cout << "\t\t\t!!++press enter to return to main menu++!!\n";
```

```
	cout << "\t\t\t\t\t";
	cin.get();
	system("cls");
}

int main() {
	Node aNode;
	bool flag = false;
	Linklist aList;
ishtart:
	system("cls");
	if (flag == true) {
		cout << "Your List:\n";
		aList.displayAll();
		cout << "Size of the list : " << aList.getSize();
		cout << endl;
	}
	cout << endl;
	cout << "\tEnter 1 to create list\n";
	cout << endl;
	cout << "\tEnter 2 to insert inside list\n";
	cout << endl;
	cout << "\tEnter 3 to delete node\n";
	cout << endl;
	cout << "\tEnter 4 to dispaly list\n";
	cout << endl;
	cout << "\tEnter 5 to swap\n";
	cout << endl;
	cout << "\tEnter 6 to copy\n";
	cout << endl;
	cout << "\tEnter 7 to update\n";
	cout << endl;
	cout << "\tEnter 8 to find values\n";
	cout << endl;
	cout << "\tEnter 9 to check length of list\n";
	cout << endl;
```

```
cout << "\tEnter 10 to delete entire lsit\n";
cout << endl;
cout << "\tEnter 11 to exit\n";

int choice;
cin >> choice;
if (choice < 1 || choice > 11) {
	cout << endl;
	cout << "\t\t\t!!!---Invalid Choice---!!!\n";
	goAhead();
	goto ishtart;
}
switch (choice) {
case 1:		//creating list
	flag = true;
	cout << "List created with name aList\n";
	goAhead();
	goto ishtart;
	break;

case 2:		//insertion
	system("cls");
	while (flag != true) {
		cout << "\t\t NO List created yet\n";
		cout << endl;
		cout << endl;
		goAhead();
		goto ishtart;
	}
	inserting:
	cout << "Your List:\n";
	aList.displayAll();
	cout << "Size of the list : " << aList.getSize();
	cout << endl << endl;
	cout << "\t\tEnter 1 to insert at start\n";
	cout << endl;
```

```
cout << "\t\tEnter 2 to insert at last\n";
cout << endl;
cout << "\t\tEnter 3 to insert at specific location\n";
cout << endl;
cout << "\t\tEnter 4 to go back to menu\n";
cout << endl;
cout << "\t\tEnter 5 to exit\n";
cout << endl;

int a;
cin >> a;
if (a < 1 || a > 5) {
	cout << endl;
	cout << "\t\t\t!!!---Invalid Choice---!!!\n";
	goAhead();
	goto inserting;
}
switch (a) {
case 1:
	int a;
	cout << "\tENTER VALUE : \n";
	cin >> a;
	aList.insertStart(a);
	goAhead();
	goto inserting;
	break;

case 2:
	int b;
	cout << "\tENTER VALUE : \n";
	cin >> b;
	aList.insertEnd(b);
	goAhead();
	goto inserting;
	break;
```

```
        case 3:
                int c;
                cout << "\tENTER VALUE : \n";
                cin >> c;
                int d;
                cout << "\tENTER LOCATION : \n";
                cin >> d;
                aList.insertSpecific(c, d);
                goAhead();
                goto inserting;
                break;

        case 4:
                goto ishtart;
                break;

        case 5:
                exit(0);
                break;
        }
        break;



case 3:         //deletion
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goAhead();
                goto ishtart;
        }
deleting:
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
```

```
cout << endl << endl;
cout << endl;
cout << "\t\tEnter 1 to delete from start\n";
cout << endl;
cout << "\t\tEnter 2 to delete from last\n";
cout << endl;
cout << "\t\tEnter 3 to delete from specific location\n";
cout << endl;
cout << "\t\tEnter 4 to go back to menu\n";
cout << endl;
cout << "\t\tEnter 5 to exit\n";
cout << endl;

int b;
cin >> b;
if (b < 1 || b > 5) {
	cout << endl;
	cout << "\t\t\t!!!---Invalid Choice---!!!\n";
	goAhead();
	goto deleting;
}
switch (b) {
case 1:
	aList.deleteStart();
	goAhead();
	goto deleting;
	break;

case 2:
	aList.deleteEnd();
	goAhead();
	goto deleting;
	break;

case 3:
	int x;
```

```
                cout << "\tENTER LOCATION : \n";
                cin >> x;
                aList.deleteSpecific(x);
                goAhead();
                goto deleting;
                break;

        case 4:
                goto ishtart;
                break;

        case 5:
                exit(0);
                break;
        }
        break;


case 4:               //display
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
display:
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
        cout << endl << endl;
        cout << endl;
        cout << "\t\tEnter 1 to display all values\n";
        cout << endl;
        cout << "\t\tEnter 2 to display odd values\n";
        cout << endl;
```

```
cout << "\t\tEnter 3 to display even values\n";
cout << endl;
cout << "\t\tEnter 4 to return to main menu\n";
cout << endl;
cout << "\t\tEnter 5 to exit\n";
cout << endl;

int q;
cin>>q;
switch (q) {
case 1:
        cout << "\t\tYour List:\n";
        aList.displayAll();
        cout << endl << endl;
        goAhead();
        goto display;

case 2:
        cout << "\t\tOdd values:\n";
        aList.displayOdd();
        cout << endl << endl;
        goAhead();
        goto display;

case 3:
        cout << "\t\tEven values:\n";
        aList.displayEven();
        cout << endl << endl;
        goAhead();
        goto display;

case 4:
        goto ishtart;
        break;

case 5:
```

```
                exit(0);
                break;
        }

case 5:                 //swaping
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
swaping:
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
        cout << endl << endl;
        cout << endl;
        cout << "\t\tEnter 1 to swap by values\n";
        cout << endl;
        cout << "\t\tEnter 2 to swap by location\n";
        cout << endl;
        cout << "\t\tEnter 3 to go back to menu\n";
        cout << endl;
        cout << "\t\tEnter 4 to exit\n";
        cout << endl;

        int z;
        cin >> z;
        switch (z) {
        case 1:
                int o;
                int p;
                cout << endl;
                cout << "\t\tEnter 1st value:\n";
                cin >> o;
```

```
                cout << "\t\tEnter 2nd value:\n";
                cin >> p;
                aList.swapValue(o,p);
                goAhead();
                goto swaping;
                break;

        case 2:
                int l1;
                int l2;
                cout << "\t\tEnter 1st location\n";
                cin >> l1;
                cout << "\t\tEnter 2nd location\n";
                cin >> l2;
                aList.swapLocation(l1, l2);
                goAhead();
                goto swaping;
                break;

        case 3:
                goto ishtart;
                break;

        case 4:
                exit(0);
                break;
        }
case 6:               //copying
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
copying:
```

```
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
        cout << endl << endl;
        cout << endl;
        cout << "\t\tEnter 1 to copy using locations\n";
        cout << endl;
        cout << "\t\tEnter 2 to return to main menu\n";
        cout << endl;
        cout << "\t\tEnter 3 to exit\n";
        cout << endl;
        int k;
        cin >> k;
        switch (k) {
        case 1:
               int L1;
               int L2;
               cout << endl;
               cout << "\t\tEnter locaiton from where value is to be copied:\n";
               cin >> L1;
               cout << "\t\tEnter location to which value is copied:\n";
               cin >> L2;
               aList.copyLoc(L1, L2);
               goAhead();
               goto copying;
               break;
        case 2:
               goto ishtart;
               break;
        case 3:
               exit(0);
               break;
        }
case 7:               //update
        system("cls");
        while (flag != true) {
```

```
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
update:
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
        cout << endl << endl;
        cout << endl;
        cout << "\t\tEnter 1 to update value\n";
        cout << endl;
        cout << "\t\tEnter 2 to return to main menu\n";
        cout << endl;
        cout << "\t\tEnter 3 to exit\n";
        cout << endl;

        int y;
        cin >> y;
        switch (y) {
        case 1:
                int loca;
                int valu;
                cout << "\t\tEnter location:\n";
                cin >> loca;
                cout << "\t\tEnter new value:\n";
                cin >> valu;
                aList.update(valu, loca);
                goAhead();
                goto update;
                break;

        case 2:
                goto ishtart;
                break;
```

```
        case 3:
                exit(0);
                break;
        }
case 8:                 //Find Values
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
find:
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
        cout << endl << endl;
        cout << endl;
        cout << "\t\tEnter 1 to find value in list\n";
        cout << endl;
        cout << "\t\tEnter 2 to return to main menu\n";
        cout << endl;
        cout << "\t\tEnter 3 to exit\n";
        cout << endl;

        int f;
        cin >> f;
        switch (f) {
        case 1:
                int valu1;
                cout << "\t\tEnter value:\n";
                cin >> valu1;
                aList.findVal(valu1);
                goAhead();
                goto find;
```

```
                break;


        case 2:
                goto ishtart;
                break;


        case 3:
                exit(0);
                break;
        }
case 9:
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
        cout << "Your List:\n";
        aList.displayAll();
        cout << "Size of the list : " << aList.getSize();
        cout << endl << endl;
        cout << endl;
        aList.len();
        goAhead();
        goto ishtart;


case 10:              //delete list
        system("cls");
        while (flag != true) {
                cout << "\t\t NO List created yet\n";
                cout << endl;
                cout << endl;
                goto ishtart;
        }
        flag = false;
```

```
                aList.delList();
                goAhead();
                goto ishtart;
        case 11:
                exit(0);
        }
        return 0;
}
```