PROGRAM CODE

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

typedef struct map
{
	char *key;
	char *value;
} map;

int PROGADDR;
int CSADDR;
int CSLTH;
int EXECADDR;
map ESTAB[100];
int ESTAB_SIZE = 0;

int getAddress()
{
	return 4096;
}

int containsKey(map arr[], int size, char *key)
{
	if (key == NULL)
		return 0;

	for (int i = 0; i < size; i++)
		if (!strcmp(arr[i].key, key))
			return 1;

	return 0;
}

void updateKey(map arr[], int size, char *key, char *value)
{
	if (key == NULL)
		return;

	for (int i = 0; i < size; i++)
		if (!strcmp(arr[i].key, key))
			arr[i].value = value;
}

char *getValue(map arr[], int size, char *key)
{
	if (key == NULL)
		return NULL;

	for (int i = 0; i < size; i++)
		if (!strcmp(arr[i].key, key))
		{
			char *temp = strdup(arr[i].value);
			return temp;
		}

	return NULL;
}
```

```
char *intToHex(int num)
{
	char *buff = malloc(7 * sizeof(char));
	sprintf(buff, "%X", num);
	return buff;
}

int hexToInt(char *buff)
{
	int num;
	sscanf(buff, "%X", &num);
	return num;
}

void displayObj(char *filename)
{
	FILE *fp = fopen(filename, "r");
	char buff[100];

	if (!fp)
	{
		printf("No such file: %s\n", filename);
		exit(0);
	}

	printf("\nObject program:\n");
	while (1)
	{
		fgets(buff, 100, fp);
		if (feof(fp))
		break;
		fputs(buff, stdout);
	}
	printf("\n");

	fclose(fp);
}

void displayMem()
{
	FILE *memFile = fopen("memory", "r");

	char buff[10];
	fgets(buff, 10, memFile);

	printf("\nMemory Allocation:\n");

	while (!feof(memFile))
	{
		printf("%s", buff);

		fgets(buff, 10, memFile);
	}
}

void displayESTAB()
{
	printf("ESTAB:\n");
	for (int i = 0; i < ESTAB_SIZE; i++)
	{
		fputs(ESTAB[i].key, stdout);
		fputs("\t", stdout);
		fputs(ESTAB[i].value, stdout);
		fputs("\n", stdout);
	}
}

void displayEstabFile()
{
	FILE *fp = fopen("estab", "r");

	printf("ESTAB:\n");
	if (fp)
	{
		char buff[20];

		while (1)
		{
			fgets(buff, 20, fp);
			if (feof(fp))
				break;

			fputs(strtok(buff, "\t"), stdout);
			printf("\t");

			char *temp = strtok(NULL, "\t");
			if (temp[strlen(temp) - 1] == '\n')
				temp[strlen(temp) - 1] = '\0';

			fputs(temp, stdout);
			printf("\n");
		}

		fclose(fp);
	}
}

void pass1(char *filename)
{
	FILE *objFile = fopen(filename, "r");

	PROGADDR = getAddress();
	CSADDR = PROGADDR;

	char buff[100], CSNAME[6];
	fgets(buff, 100, objFile);

	while (!feof(objFile))
	{
		if (buff[0] == 'H')
		{
			for (int i = 2; i < 8; i++)
			{
				CSNAME[i - 2] = buff[i];

				if (buff[i + 1] == ' ')
				{
					CSNAME[i - 1] = '\0';
					break;
				}
			}

			if (containsKey(ESTAB, ESTAB_SIZE, CSNAME))
			{
				printf("\nDuplicate control section name \"%s\". Aborted.\n", CSNAME);
```

```
                        fclose(objFile);
                        remove("estab");
                        exit(0);
                }
                else
                {
                        ESTAB[ESTAB_SIZE].key = strdup(CSNAME);
                        ESTAB[ESTAB_SIZE].value = intToHex(CSADDR);

                        ESTAB_SIZE++;

                        printf("Control Section Name: %s\n", CSNAME);
                }

                char temp[6];
                for (int i = 0; i < 6; i++)
                        temp[i] = buff[i + 16];
                temp[6] = '\0';

                CSLTH = hexToInt(temp);

                printf("Control Section Start Address: %X\n", CSADDR);
                printf("Control Section Length: %XH\n", CSLTH);
        }

        fgets(buff, 100, objFile);

        while (buff[0] != 'E')
        {
                char *token = strtok(buff, "^");

                if (!strcmp(token, "D"))
                {
                        while (1)
                        {
                                token = strtok(NULL, "^");

                                if (!token)
                                        break;

                                for (int i = 0; i < 6; i++)
                                        if (token[i] == ' ')
                                                token[i] = '\0';

                                if (containsKey(ESTAB, ESTAB_SIZE, token))
                                {
                                        printf("\nDuplicate external symbol \"%s\". Aborted.\n", token);
                                        fclose(objFile);
                                        remove("estab");
                                        exit(0);
                                }

                                ESTAB[ESTAB_SIZE].key = strdup(token);

                                token = strtok(NULL, "^");

                                if (!token)
                                {
                                        printf("Bad Define record. Aborted.\n");
                                        fclose(objFile);
                                        exit(0);
                                }
```

```
					ESTAB[ESTAB_SIZE].value = strdup(intToHex(hexToInt(token) +
CSADDR));

					ESTAB_SIZE++;
				}
			}

			fgets(buff, 100, objFile);
		}

		printf("\n%s pass 1 completed.\n\n", CSNAME);
		CSADDR += CSLTH;

		fgets(buff, 100, objFile);
	}

	fclose(objFile);
}

void pass2(char *filename)
{
	FILE *objFile = fopen(filename, "r");
	FILE *memFile = fopen("memory", "w+");
	int addr = PROGADDR, textLen;
	long seek1;

	CSADDR = PROGADDR;

	char buff[100], CSNAME[6];
	fgets(buff, 100, objFile);

	while (!feof(objFile))
	{
		EXECADDR = CSADDR;

		if (buff[0] == 'H')
		{
			for (int i = 2; i < 8; i++)
			{
				CSNAME[i - 2] = buff[i];

				if (buff[i + 1] == ' ')
				{
					CSNAME[i - 1] = '\0';
					break;
				}
			}

			char temp[] = {buff[9], buff[10], buff[11], buff[12], buff[13], buff[14], '\0'};

			int csAddrInObj = hexToInt(temp); //always 000000

			for (int i = 0; i < 6; i++)
				temp[i] = buff[i + 16];

			CSLTH = hexToInt(temp);
		}

		fgets(buff, 100, objFile);

		while (buff[0] != 'E')
		{
			char *token = strtok(buff, "^");
```

```
if (!strcmp(token, "T"))
{
	token = strtok(NULL, "^");

	while (addr < CSADDR + hexToInt(token))
	{
		fputs(intToHex(addr), memFile);
		fputs(": XX\n", memFile);
		addr++;
	}

	addr = CSADDR + hexToInt(token);

	token = strtok(NULL, "^");

	textLen = hexToInt(token);

	while (1)
	{
		token = strtok(NULL, "^");

		if (!token)
			break;
		else
		{
			if (token[strlen(token) - 1] == '\n')
				token[strlen(token) - 1] = '\0';

			for (int i = 0; i < strlen(token); i = i + 2)
			{
				fputs(intToHex(addr), memFile);
				fputs(": ", memFile);

				char temp[] = {token[i], token[i + 1], '\n', '\0'};
				fputs(temp, memFile);

				addr++;
			}
		}
	}
}
else if (!strcmp(token, "M"))
{
	seek1 = ftell(memFile);

	token = strtok(NULL, "^");

	int modAddr = hexToInt(token) + CSADDR;

	token = strtok(NULL, "^");

	int bytes = ceil(hexToInt(token) / 2.0);

	token = strtok(NULL, "^");

	char sign = token[0];

	char *symbol = strdup(++token);

	symbol[strlen(symbol) - 1] = '\0';

	if (containsKey(ESTAB, ESTAB_SIZE, symbol))
```

```
                                {
                                        rewind(memFile);

                                        while (!feof(memFile))
                                        {
                                                long seek = ftell(memFile);
                                                fgets(buff, 100, memFile);

                                                token = strtok(buff, ": ");

                                                if (!strcmp(token, intToHex(modAddr)))
                                                {
                                                        char newAddr[bytes * 2];

                                                        strcpy(newAddr, strtok(NULL, ": "));

                                                        newAddr[2] = '\0';

                                                        for (int i = 1; i < bytes; i++)
                                                        {
                                                                fgets(buff, 100, memFile);
                                                                token = strtok(buff, ": ");
                                                                token = strtok(NULL, ": ");
                                                                token[2] = '\0';

                                                                strcat(newAddr, token);
                                                        }

                                                        char *mod;

                                                        if(sign == '+')
                                                                mod = intToHex(hexToInt(newAddr) +
hexToInt(getValue(ESTAB, ESTAB_SIZE, symbol)));
                                                        else
                                                                mod = intToHex(hexToInt(newAddr) -
hexToInt(getValue(ESTAB, ESTAB_SIZE, symbol)));

                                                        int len = strlen(mod);

                                                        char temp[bytes * 2];
                                                        strcpy(temp, "0");

                                                        for (int i = bytes * 2; i > len; i--)
                                                        {
                                                                strcat(temp, mod);
                                                                strcpy(mod, temp);
                                                                strcpy(temp, "0");
                                                        }

                                                        fseek(memFile, seek, SEEK_SET);

                                                        for (int i = 0; i < bytes * 2; i = i + 2, modAddr++)
                                                        {
                                                                char modStr[] = {mod[i], mod[i + 1], '\n', '\0'};
                                                                fputs(intToHex(modAddr), memFile);
                                                                fputs(": ", memFile);
                                                                fputs(modStr, memFile);
                                                        }

                                                        break;
                                                }
                                        }
                                }
```

```
                    else
                    {
                        printf("\nUndefined external symbol \"%s\". Aborted.\n", symbol);
                        fclose(objFile);
                        fclose(memFile);
                        remove("estab");
                        exit(0);
                    }

                    fseek(memFile, seek1, SEEK_SET);
                }

                fgets(buff, 100, objFile);
            }

            while (addr <= CSADDR + CSLTH)
            {
                fputs(intToHex(addr), memFile);
                fputs(": XX\n", memFile);
                addr++;
            }

            strtok(buff, "^");

            if(!strtok(NULL, "^")) {
                char temp[] = {buff[2], buff[3], buff[4], buff[5], buff[6], buff[7], '\0'};

                EXECADDR = CSADDR + hexToInt(temp);
            }

            CSADDR += CSLTH;

            fgets(buff, 100, objFile);

            printf("\n%s pass 2 completed.\n", CSNAME);

            printf("\nExecution of %s starts at %XH\n", CSNAME, EXECADDR);
        }

        fclose(objFile);
        fclose(memFile);
        //remove("estab");
}

void initialise()
{
        FILE *fp = fopen("estab", "r");

        if (fp)
        {
            char buff[20];

            while (1)
            {
                fgets(buff, 20, fp);

                if (feof(fp))
                    break;

                ESTAB[ESTAB_SIZE].key = strdup(strtok(buff, "\t"));

                char *temp = strtok(NULL, "\t");
```

```
			if (temp[strlen(temp) - 1] == '\n')
				temp[strlen(temp) - 1] = '\0';

			ESTAB[ESTAB_SIZE].value = strdup(temp);

			ESTAB_SIZE++;
		}

		fclose(fp);
	}
}

void writeESTAB()
{
	FILE *fp = fopen("estab", "w");

	for (int i = 0; i < ESTAB_SIZE; i++)
	{
		fputs(ESTAB[i].key, fp);
		fputs("\t", fp);
		fputs(ESTAB[i].value, fp);
		fputs("\n", fp);
	}

	fclose(fp);
}

void main()
{
	char *filename = malloc(20 * sizeof(char));

	printf("Enter filename of object program: ");
	fgets(filename, 20, stdin);

	for (int i = 0; i < 20; i++)
		if (filename[i] == '\n')
			filename[i] = '\0';

	initialise();

	displayObj(filename);

	pass1(filename);

	writeESTAB();

	displayEstabFile();

	pass2(filename);

	displayMem();
}
```

OUTPUT

```
Enter filename of object program: obj

Object program:
H^PG1   ^000000^000072
```

```
D^EXT1  ^000000^EXT2  ^000003^EXT3  ^000006^EXT123^000009
T^000000^1E^041042^10105D^10106F^00904E^40104B^28104B^301018^3C102D^04105D^00904E
T^00001E^1E^04106F^0C906C^00106F^181045^0C106F^00105D^181045^0C105D^04105D^281048
T^00003C^12^381009^4C0000^000000^000003^00000F^800000
M^000001^03^+EXT5
E^000000
H^PG2   ^000000^000072
D^EXT4  ^000072^EXT5  ^000000^EXT6  ^000078^EXT456^00007B
T^000000^1E^041042^10105D^10106F^00904E^40104B^28104B^301018^3C102D^04105D^00904E
T^00001E^1E^04106F^0C906C^00106F^181045^0C106F^00105D^181045^0C105D^04105D^281048
T^00003C^12^381009^4C0000^000000^000003^00000F^800000
M^000001^03^+EXT1
M^000004^04^-EXT2
E^

Control Section Name: PG1
Control Section Start Address: 1000
Control Section Length: 72H

PG1 pass 1 completed.

Control Section Name: PG2
Control Section Start Address: 1072
Control Section Length: 72H

PG2 pass 1 completed.

ESTAB:
PG1     1000
EXT1    1000
EXT2    1003
EXT3    1006
EXT123  1009
PG2     1072
EXT4    10E4
EXT5    1072
EXT6    10EA
EXT456  10ED

PG1 pass 2 completed.

Execution of PG1 starts at 1000H

PG2 pass 2 completed.

Execution of PG2 starts at 1072H

Memory Allocation:
1000: 04
1001: 20
1002: B4
1003: 10
1004: 10
1005: 5D
1006: 10
1007: 10
1008: 6F
1009: 00
100A: 90
100B: 4E
100C: 40
100D: 10
100E: 4B
```

```
100F: 28
1010: 10
1011: 4B
1012: 30
1013: 10
1014: 18
1015: 3C
1016: 10
1017: 2D
1018: 04
1019: 10
101A: 5D
101B: 00
101C: 90
101D: 4E
101E: 04
101F: 10
1020: 6F
1021: 0C
1022: 90
1023: 6C
1024: 00
1025: 10
1026: 6F
1027: 18
1028: 10
1029: 45
102A: 0C
102B: 10
102C: 6F
102D: 00
102E: 10
102F: 5D
1030: 18
1031: 10
1032: 45
1033: 0C
1034: 10
1035: 5D
1036: 04
1037: 10
1038: 5D
1039: 28
103A: 10
103B: 48
103C: 38
103D: 10
103E: 09
103F: 4C
1040: 00
1041: 00
1042: 00
1043: 00
1044: 00
1045: 00
1046: 00
1047: 03
1048: 00
1049: 00
104A: 0F
104B: 80
104C: 00
104D: 00
```

```
104E: XX
104F: XX
1050: XX
1051: XX
1052: XX
1053: XX
1054: XX
1055: XX
1056: XX
1057: XX
1058: XX
1059: XX
105A: XX
105B: XX
105C: XX
105D: XX
105E: XX
105F: XX
1060: XX
1061: XX
1062: XX
1063: XX
1064: XX
1065: XX
1066: XX
1067: XX
1068: XX
1069: XX
106A: XX
106B: XX
106C: XX
106D: XX
106E: XX
106F: XX
1070: XX
1071: XX
1072: XX
1072: 04
1073: 20
1074: 42
1075: 10
1076: 00
1077: 5A
1078: 10
1079: 10
107A: 6F
107B: 00
107C: 90
107D: 4E
107E: 40
107F: 10
1080: 4B
1081: 28
1082: 10
1083: 4B
1084: 30
1085: 10
1086: 18
1087: 3C
1088: 10
1089: 2D
108A: 04
108B: 10
```

```
108C: 5D
108D: 00
108E: 90
108F: 4E
1090: 04
1091: 10
1092: 6F
1093: 0C
1094: 90
1095: 6C
1096: 00
1097: 10
1098: 6F
1099: 18
109A: 10
109B: 45
109C: 0C
109D: 10
109E: 6F
109F: 00
10A0: 10
10A1: 5D
10A2: 18
10A3: 10
10A4: 45
10A5: 0C
10A6: 10
10A7: 5D
10A8: 04
10A9: 10
10AA: 5D
10AB: 28
10AC: 10
10AD: 48
10AE: 38
10AF: 10
10B0: 09
10B1: 4C
10B2: 00
10B3: 00
10B4: 00
10B5: 00
10B6: 00
10B7: 00
10B8: 00
10B9: 03
10BA: 00
10BB: 00
10BC: 0F
10BD: 80
10BE: 00
10BF: 00
10C0: XX
10C1: XX
10C2: XX
10C3: XX
10C4: XX
10C5: XX
10C6: XX
10C7: XX
10C8: XX
10C9: XX
10CA: XX
```

```
10CB: XX
10CC: XX
10CD: XX
10CE: XX
10CF: XX
10D0: XX
10D1: XX
10D2: XX
10D3: XX
10D4: XX
10D5: XX
10D6: XX
10D7: XX
10D8: XX
10D9: XX
10DA: XX
10DB: XX
10DC: XX
10DD: XX
10DE: XX
10DF: XX
10E0: XX
10E1: XX
10E2: XX
10E3: XX
10E4: XX
```