Structures – Q1 (computeExp)

(**computeExp**) A structure is defined to represent an arithmetic expression:

```
typedef struct {
        float operand1, operand2;
        char op;      /* operator '+','-','*' or '/'  */
} bexpression;
```

(a) Write a C function that computes the value of an expression and returns the result. For example, the function will return the value of 4/2 if in the structure passed to it, operand1 is 4, operator is '/' and operand2 is 2. The function prototype is given as:

float compute1(bexpression *expr*);

(b) Write another C function that performs the same computation with the following function prototype:

float compute2(bexpression **expr*);

Write a C program to test the functions.

Sample input and output sessions:

(1) Test Case 1
Enter expression (op1 op2 op) :
4 8 +
compute1 = 12.00
compute2 = 12.00

(2) Test Case 2
Enter expression (op1 op2 op) :
8 4 /
compute1 = 2.00
compute2 = 2.00

(3) Test Case 3
Enter expression (op1 op2 op) :
4 8 *
compute1 = 32.00
compute2 = 32.00

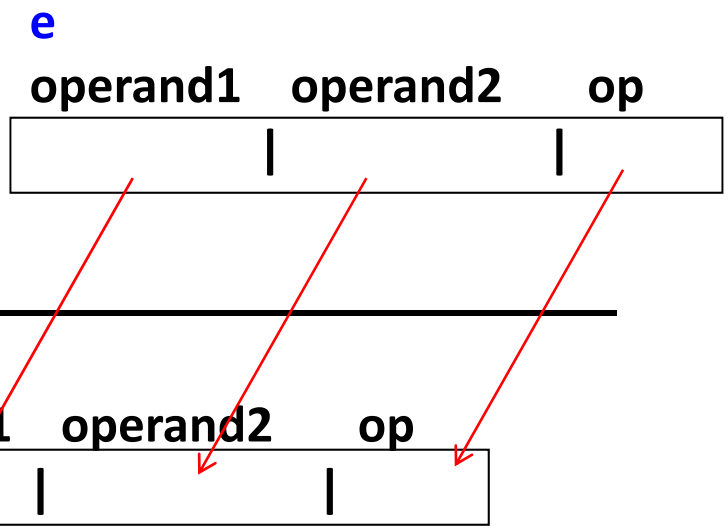
Structures – Q1 (computeExp)

```
#include <stdio.h>
typedef struct {
   float operand1, operand2;
   char op;
} bexpression;
float compute1(bexpression expr);
float compute2(bexpression *expr);
int main()
{
   bexpression e;
   int choice;
   printf("Select one of the following options: \n");
   printf("1: compute1()\n");
   printf("2: compute2()\n");
   printf("3: exit()\n");
   do {
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            printf("Enter expression (op1 op2 op): \n");
            scanf("%f %f %c", &e.operand1, &e.operand2, &e.op);
            printf("compute1(): %.2f\n", compute1(e));
            break;
         case 2:
            printf("Enter expression (op1 op2 op): \n");
            scanf("%f %f %c", &e.operand1, &e.operand2, &e.op);
            printf("compute2(): %.2f\n", compute2(&e));
            break;
      }
   } while (choice < 3);
   return 0; }
```

Structures – Q1 (computeExp)

```
int main()
{
   printf("compute1(): %.2f\n", compute1(e));


}
```




```
float compute1(bexpression expr){
    float result;
    switch (expr.op) {
       case '+': result = expr.operand1 + expr.operand2;
          break;
       case '-': result = expr.operand1 - expr.operand2;
          break;
       case '*': result = expr.operand1 * expr.operand2;
          break;
       case '/': result = expr.operand1 / expr.operand2;
          break;
    }
    return result;
}
```

Use dot notation when accessing members of the structure in this function.

Structures – Q1 (computeExp)

```
int main()
{
      printf("compute2(): %.2f\n", compute2(&e));

}
```

Call by reference


```
float compute2(bexpression *expr)
{
   float result;
   switch (expr->op) {
      case '+': result = expr->operand1 + expr->operand2;
         break;
      case '-': result = expr->operand1 - expr->operand2;
         break;
      case '*': result = expr->operand1 * expr->operand2;
         break;
      case '/': result = expr->operand1 / expr->operand2;
         break;
   }
   return result;
}
```

Use -> notation when accessing members of the structure in this function.

Structures – Q2 (phoneBook)

Write a C program that implements the following three functions:

- The function readin() reads a number of persons' names and their corresponding telephone numbers, passes the data to the caller via the parameter p, and returns the number of names that have entered. The character '#' is used to indicate the end of user input.
- The function printPB() prints the phonebook information on the display. It will print the message "Empty phonebook" if the phonebook list is empty.
- The function search() finds the telephone number of an input name *target*, and then prints the name and telephone number on the screen. If the input name cannot be found, then it will print an appropriate error message. The prototypes of the two functions are given below:

The prototypes of the three functions are given below:

```
void printPB(PhoneBk *pb, int size);
int readin(PhoneBk *pb);
void search(PhoneBk *pb, int size, char *target);
```

```
#include <stdio.h>
#include <string.h>
#define MAX 100
typedef struct {
  char name[20];
  char telno[20];
} PhoneBk;
void printPB(PhoneBk *pb, int size);
int readin(PhoneBk *pb);
void search(PhoneBk *pb, int size, char *target);
int main()
{
  PhoneBk s[MAX];
  char t[20], *p;
  int size=0, choice, dummychar;

  printf("Select one of the following options: \n");
  printf("1: readin()\n");
  printf("2: search()\n");
  printf("3: printPB()\n");
  printf("4: exit()\n");
  do {
    printf("Enter your choice: \n");
    scanf("%d", &choice);
```

```
switch (choice) {
      case 1:
        scanf("%c", &dummychar);
        size = readin(s);
        break;
      case 2:
        scanf("%c", &dummychar);
        printf("Enter search name: \n");
        fgets(t, 20, stdin);
        if (p=strchr(t,'\n')) *p = '\0';
        search(s,size,t);
        break;
      case 3:
        printPB(s, size);
        break;
    }
  } while (choice < 4);
  return 0;
}
```

```
void printPB(PhoneBk *pb, int size)
{
  int i;

  printf("The phonebook list: \n");
  if (size==0)
    printf("Empty phonebook\n");
  else {
    for (i=0; i<size; i++) {
      printf("Name: %s\n", (pb+i)->name);
      printf("Telno: %s\n", (pb+i)->telno);
    }
  }
}
```

```
int readin(PhoneBk *pb)
{
  int size=0;
  char *p;

  while (1) {
    printf("Enter name: \n");
    fgets(pb->name, 80, stdin);
    if (p=strchr(pb->name,'\n')) *p = '\0';
    if (strcmp(pb->name,"#")==0)
      break;
    printf("Enter tel: \n");
    fgets(pb->telno, 80, stdin);
    if (p=strchr(pb->telno,'\n')) *p = '\0';
    pb++;
    size++;
  }
  return size;
}
```

```
void search(PhoneBk *pb, int size, char *target)
{
  int i;

  for (i=0;i<size;i++,pb++){
    if (strcmp(pb->name,target)==0){
      printf("Name = %s, Tel = %s\n",target,pb->telno);
      break;
    }
  }
  if (i==size)
    printf("Name not found!\n");
}
```